VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Vícefaktorová autentizace v embedded zařízeních typu Raspberry Pi

Multi-factor Authentication in Embedded Systems

Bc. Zdeněk Kalich

Diplomová práce

Vedoucí práce: Ing. Pavel Nevlud

Ostrava, 2021

Abstrakt

Tato diplomová práce se zabývá návrhem, implementací a testováním vícefaktorové autentizace v embedded zařízení Raspberry Pi 4 Model B. V teoretické části jsou představeny jednotlivé autentizační metody, vysvětlena vícefaktorová autentizace a popsána struktura a možnosti autentizace pomoci PAM v operačním systému Linux. Je zde také představeno několik dalších embedded zařízení typu Raspberry Pi. Praktická část je rozdělena na dvě částí. V první z těchto částí je navrženo a otestováno několik variant vícefaktorové autentizace pro Raspberry Pi v režimu klasického stolního počítače s připojením všech periferních zařízení. Ve druhé části je navrženo a otestováno několik variant vícefaktorové autentizace pro vzdálený přístup k zařízení Raspberry Pi pomoci SSH nebo SFTP.

Klíčová slova

vícefaktorová autentizace, MFA, embedded, Raspberry Pi, Linux, PAM, RFID, biometie, YubiKey, Google Authenticator, SSH, SFTP

Abstract

This diploma thesis deals with the design, implementation and testing of multifactor authentication in an embedded Raspberry Pi 4 Model B device. The theoretical part introduces the various authentication methods, explains the multifactor authentication and describes the structure and possibilities of PAM authentication in the Linux operating system. Several other embedded of Raspberry Pi type devices are also introduced. The practical part is divided into two parts. In the first of these parts, several variants of multifactor authentication for Raspberry Pi in the mode of a classic desktop computer with connection of all peripheral devices are designed and tested. In the second part, several variants of multifactor authentication for remote access to Raspberry Pi devices using SSH or SFTP are designed and tested.

Keywords

multi-factor authentication, MFA, embedded, Raspberry Pi, Linux, PAM, RFID, biometrics, YubiKey, Google Authenticator, SSH, SFTP

Poděkování

Rád bych na tomto místě poděkoval Ing. Pavlu Nevludovi, vedoucímu mé diplomové práce, za ochotu a poskytnutí spousty užitečných rad, potřebných pro vypracování mé práce.

Obsah

Seznam použitých zkratek a symbolů

ARM	–	Advanced RISC Machines
CMOS	–	Complementary Metal–Oxide–Semiconductor
ECC	–	Elliptical Curve Cryptography
eMMC	–	embedded Multi-Media Card
GPIO	–	General-Purpose Input/Output
GPU	–	Graphics Processing Unit
HDMI	–	High Definition Multimedia Interface
HF	–	High-Frequency
HID	–	Human Interface Device
HMAC	–	Keyed-hash Message Authentication Code
HOTP	–	HMAC-based One-time Password Algorithm
IP	–	Internet Protocol
I2C	–	Inter Integrated Circuit
LF	–	Low-Frequency
LPDDR	–	Low-Power Double Data Rate
MD5	–	Message-Digest algorithm
MFA	–	Multi-factor Authentication
MitM	–	Man in the Middle
NFC	–	Near Field Communication
OS	–	Operating System
OTG	–	On The Go
OTP	–	One Time Password
PAM	–	Pluggable Authentication Modules
PIN	–	Personal Identification Number
PoE	–	Power over Ethernet
QR	–	Quick Response
RAM	–	Random Access Memory
RFID	–	Radio Frequency Identification

RSA	–	Rivest Shamir Adleman
SD	–	Secure Digital
SFTP	–	SSH File Transfer Protocol
SHA	–	Secure Hash Algorithm
SMS	–	Short Message Service
SPI	–	Serial Peripheral Interface
SSH	–	Secure Shell
TOTP	–	Time-based One-time Password Algorithm
UART	–	Universal Asynchronous Receiver Transmitter
UHF	–	Ultra-High Frequency
USB	–	Universal Serial Bus
U2F	–	Universal 2nd Factor
VPN	–	Virtual Private Network
WiFI	–	Wireless Fidelity Universal
2FA	–	Two Factor Authentication

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

S autentizací se v počítači nebo mobilním telefonu setkal určitě každý z nás a je také možné, že tímto ověřovacím procesem prochází denně. Někdo nemusí tušit, co je za pojmem autentizace skryto, ale když mu vysvětlíte, že autentizace je proces ověření identity uživatele, například při přihlášení k e-mailovému účtu pomoci e-mailové adresy a jeho tajného hesla, hned pochopí o co se jedná. Pojem vícefaktorová autentizace, však už může být oříšek i pro zvídavější uživatelé, kteří si často myslí, že ví, co tento pojem znamená, ale nerozlišují mezi vícekrokovou a vícefaktorovou autentizaci.

V této pokročilé době, plné počítačů a nových technologií, kde nebezpečí číhá na každém kroku, si lidé a firmy uvědomují možná rizika a chtějí lépe chránit svá zařízení, účet nebo síť, se můžeme s vícefaktorovou autentizací potkávat mnohem častěji. Využití vícefaktorové autentizace můžeme pozorovat u zaměstnanců bank a korporaci, kdy zaměstnanci pro přihlášení ke svému pracovnímu účtu využívají vícefaktorovou autentizaci, nejčastěji se jedná o uživatelské heslo a kód OTP, který je generován pomoci aplikace v mobilním telefonu.

Tato diplomová práce se zabývá možnosti implementace vícefaktorové autentizace na embedded zařízeních typu Raspberry Pi, konkrétně na zařízení Raspberry Pi 4 Model B, kde jsou navrženy a otestovány některé možné způsoby vícefaktorové autentizace.

Druhá kapitola se zabývá vysvětlením pojmu autentizace a popisem jednotlivých autentizačních metod.

Ve třetí kapitole je vysvětlen pojem vícefaktorová autentizace a další pojmy nebo náležitosti s tím spojené.

Čtvrtá kapitola popisuje možnosti autentizace pomoci PAM v operačním systému Linux, jako je struktura PAM, princip fungování nebo popis jednotlivých autentizačních modulů a syntaxe konfiguračních souborů.

Pátá kapitola objasňuje pojem embedded zařízení a popisuje základní parametry a vlastnosti zařízení Raspberry Pi 4 Model B, včetně představení několika dalších embedded zařízení tohoto typu, které jsou na trhu k dispozici.

Šestá kapitola se zabývá návrhem vícefaktorové autentizace pro zařízení Raspberry Pi, kdy toto zařízení využíváme především jako klasický počítač a máme k němu připojeny všechny periferní zařízení. Je zde podrobně popsán postup implementace tří variant vícefaktorové autentizace a nastíněny další možnosti a varianty MFA.

Sedmá kapitola se zabývá implementací třech variant vícefaktorové autentizace pro vzdálený přístup k zařízení Raspberry Pi pomoci SSH a SFTP.

Kapitola 2

Možnosti autentizace

Autentizace je zabezpečený proces zjištění a ověření identity subjektu, který požaduje přístup k systému, síti nebo zařízení. Subjektem může být uživatel, program nebo zařízení. Jde o ověření, zda je daný subjekt opravdu tím, za koho se vydává. [1] Autentizace brání neoprávněným subjektům v přístupu k citlivým informacím nebo obecně do míst, kam by neměli mít přístup. Bez zabezpečeného procesu autentizace může být systém, zařízení nebo síť v ohrožení.

Jsou různé metody, jak lze subjekt autentizovat, ale existují pouze tři základní kategorie, do kterých by měla patřit každá z těchto metod. Tyto kategorie jsou:

- Znalost - něco vím (heslo, PIN)
- Vlastnictví - něco mám (čipová karta, hardwarový token)
- Biometrické vlastnosti, existence - něco jsem (otisk prstu)

2.1 Autentizace znalostí

Metoda je založena na znalosti tajné informace, kterou ostatní uživatele nemají k dispozici, ale zná ji pouze oprávněná osoba. Typickým příkladem autentizace znalostí je kombinace uživatelského jména a hesla, PIN kód nebo odpověď na kontrolní otázku. Pro tento způsob autentizace postačuje pouze databáze uživatelských hesel, proti které se porovnají informace získané od uživatele. [1] Výhodou autentizace znalostí je fakt, že je snadná, levná a uživatel nepotřebuje žádné dodatečné vybavení. Tato tajná informace, jako je heslo nebo PIN, však může snadno uniknout a případný únik nemusí být snadné odhalit. [2] Nebo tuto tajnou informaci můžeme zapomenout. Dalším bezpečnostním rizikem je skutečnost, že si uživatelé často volí jednoduchá hesla nebo používají stejné heslo pro přístupu k více službám, aby si ho lépe zapamatovali.

Když víme, že uživatelé mohou být při vytváření a používání hesel nezodpovědní, může alespoň druhá strana, která je zodpovědná za uložená hesla v databází, zabezpečit tuto databázi pomoci bezpečnostních mechanismů proti případnému útočníkovi.

2.1.1 Hesla ve formě prostého textu

Je to nejjednodušší možný způsob, jak ukládat hesla. V případě, že uživatel heslo zapomene, může ho snadno získat od správce, když se však útočník dostane dovnitř systému, jsou všechna hesla v ohrožení. Ukládání hesel ve formě prostého textu bychom se tedy měli snažit vyhnout. [2]

2.1.2 Hesla v šifrované podobě

Heslo je ve větším bezpečí, když je kryptograficky chráněno a to například s využitím blokových šifer. Hesla uložená v této podobě jsou poměrně odolná vůči útokům, ale neumožňují obnovení ztraceného hesla. [2]

2.1.3 Hesla v hashované podobě

Hesla by se měla do databáze ukládat nejlépe v hashované podobě, kdy vezmeme původní heslo, to projde hashovací funkcí a vytvoří se tak jeho hash (otisk). Tyto hashovací funkce se označují jako jednocestné a původní heslo nelze z otisku zpětně spočítat. Může se jednat například o hashovací funkci MD5 nebo SHA-256. [3]

2.2 Autentizace vlastnictvím

Tento způsob autentizace spočívá v tom, že uživatel pro autentizaci použije unikátní předmět, který vlastní a má ho u sebe. Jedná se většinou o mobilní telefon, bezkontaktní kartu, přívěsek a softwarový nebo hardwarový token. [1] Takovéto předměty můžeme poměrně lehce ztratit nebo nám mohou být odcizeny, výhoda však spočívá v tom, že si toho lze snadno všimnout. Tyto předměty jsou většinou těžko kopírovatelné. Z pohledu periferních zařízení se může připojená čtečka karet nebo hardwarový token někdy jevit jako překážka a omezení.

2.2.1 Radio Frequency Identification (RFID)

Je bezdrátová technologie pro identifikaci pomoci dat, přenášených prostřednictvím rádiových vln. Informace je uložena do tzn. RFID tagu, který můžeme načítat nebo i přepisovat. RFID tag může být zakomponován do předmětů různých tvarů, barev a velikostí. Tyto RFID tagy lze číst nebo přepisovat pomoci RFID čtečky. [4] RFID čtečky pracují na třech hlavních frekvencích

- 125 kHz (pásmo LF)
- 13,56 MHz (HF pásmo)
- 868 MHz (pásmo UHF)

Tento parametr frekvence spolu s velikostí antény má velký vliv na čtecí vzdálenost. S čtečkou v pásmu UHF lze dosáhnout čtecí vzdálenosti až několik metrů a využívají se většinou pro systémy řízení skladů. Zato typická vzdálenost čtení pro čtečky v pásmu LF a HF je přibližně 3-15 cm a používají se zejména pro sledování docházky a identifikaci osob. [5]

2.2.2 Google Authenticator

Je softwarový nástroj vytvořený společností Google, který slouží pro generování jednorázových kódů. Aplikace je dostupná na mobilních zařízeních Android, BlackBerry a iOS. Google Authenticator pro generování jednorázových hesel využívá algoritmy Time-based One-time Password Algorithm (TOTP) a HMAC-based One-time Password Algorithm (HOTP). Google poskytuje také modul PAM pro autentizaci uživatelů v systémech Linux. Díky tomu můžeme nastavit například vícefaktorovou autentizaci pro SSH nebo další služby. [6]

2.2.3 YubiKey 5

Je hardwarový token, který zajišťuje silnou hardwarovou autentizaci pro online služby, aplikace a VPN. YubiKey se vzhledově podobá obyčejnému USB flash disku. Nepotřebuje žádný speciální software ani baterii, stačí ho pouze připojit přes USB konektor nebo bezdrátově pomoci NFC a MIFARE a při použití se dotknout prstem kovového kontaktu, pro přenesení malého elektrického náboje a vygenerování kódu. Tímto kontaktem zároveň dojde k ověření, že jste člověk a ne vzdálený útočník. Má podporu v operačních systémech Microsoft Windows, MacOS, Linux a Android. YubiKey je víceprotokolový bezpečnostní klíč, který podporuje protokoly jako je OTP, Smart Card, OpenPGP, OATH, FIDO U2F nebo FIDO2 pro přihlášení bez hesla. [7] Některé z těchto protokolů jsou více popsány níže. Typ konektoru a podpora jednotlivých protokolů může být jiná u různých variant provedení tohoto hardwarového tokenu.

Obrázek 2.1: YubiKey 5 NFC [8]

2.2.3.1 OTP

OTP je jednoduchý, ale silný autentizační mechanismus. Při autentizaci je zadán jednorázový 44 místný kód. Prvních 12 znaků je konstantních a udává veřejné ID zařízení. Zbylých 32 znaků se mění a představuje jedinečný přístupový kód. Případné okopírování OTP kódu neumožňuje YubiKey zfalšovat, protože každý OTP kód funguje pouze jednou a při každém použití je generován kód nový. YubiKey komunikuje s počítačem pomoci rozhraní klávesnice HID a výstupem je sekvence stisku kláves. Protokol OTP funguje ve všech systémech, kde je podporována klávesnice USB. [7, 9]

2.2.3.2 FIDO U2F

Universal 2nd Factor je protokol vytvořený společností Google a Yubico s přispěním NXP. Protokol U2F se silně rozšířil a využívají ho služby jako Facebook, Gmail, DropBox nebo GitHub. Umožňuje uživatelům internetu bezpečný přístup k libovolnému počtu online služeb pomoci jediného bezpečnostního klíče. Zároveň poskytuje ochranu proti útoku MitM a phishingu, protože pomoci bezpečnostního klíče se může uživatel přihlásit pouze na skutečný, originální web a ověření na falešném webu selže, když by byl uživatel náhodou oklamán. [10]

2.2.3.3 OATH

OATH specifikuje dva otevřené standardy autentizace TOTP a HOTP. Při TOTP zadává uživatel 6-8 místný kód, který se mění každých 30 sekund. Kód je generován pomoci HMAC algoritmu, kde se časové razítko mění každých 30 sekund. HOTP funguje podobně, ale místo časového razítka je použit čítač ověření. Pokud je tedy vygenerován OTP kód bez toho, aby se ověřil, čítač zařízení již nebude odpovídat počítadlu serveru. HOTP kód lze vygenerovat při stisknutí tlačítka na Yubikey. Pro TOTP je potřeba ještě aplikaci, která je schopna číst kódy z YubiKey, protože samotný YubiKey nemá vnitřní hodiny. [11]

2.2.3.4 OpenPGP

Je otevřený standard pro provádění operace podepisování a šifrování RSA nebo ECC pomoci soukromého klíče, uloženého na čipové kartě, prostřednictvím rozhraní jako je PKCS#11 a Smart Card Minidriver. Standard pro šifrování e-mailu. Nepoužívá se pro webovou autentizaci. [12]

2.3 Autentizace biometrií

Tento faktor autentizace je založen na tom, že se totožnost uživatele ověřuje prostřednictvím jedinečných biologických a behaviorálních charakteristik, jako je otisk prstu, charakteristika obličeje, oční duhovka, sítnice, hlas, chůze. Jelikož jsou tyto vlastnosti pro každého uživatele jedinečné, je bio-

metrické ověřování považováno za obecně bezpečnější. Ověření uživatele probíhá tak, že biometrické senzory porovnávají biometrické vlastnosti uživatele s daty uloženými v databázi. [13]

Autentizace pomoci otisku prstu patří mezi nejoblíbenější biometrická ověření. Snímače otisku prstu však mohou pracovat na různém principu a můžeme je rozdělit do následujících skupin.

- Optické skenery - Představují nejstarší způsob získání otisku prstu, kde je otisk prstu zachycen pomoci fotografie. Následně jsou použity algoritmy k detekci vzorů na povrchu prstu, tedy k označení prohlubní a hřebenů. Tento systém lze poměrně snadno oklamat pomoci vytištěné fotografie otisku prstu.

- Kapacitní skenery - Využívají se zde malé kondenzátory. Při přiložení prstu se změní elektrický náboj kondenzátoru na hřebenu, zatímco na prohlubni zůstane náboj nezměněn. Tyto informace jsou pak poslány k dalšímu zpracování. Čím více kondenzátoru snímač obsahuje, tím přesnější otisk bude.

- Ultrazvukové skenery - Skládají se z ultrazvukového vysílače a přijímače. Ultrazvukový signál je vyslán směrem k prstu, který je přiložen na panelu skeneru. Otisk prstu je vyhodnocen na základě doby návratu odraženého signálu od prstu zpět k přijímači, kdy signál odražený od hřebene dorazí k přijímači dříve, než signál odražený na prohlubní (větší vzdálenost). [14]

Níže jsou vypsány další možnosti biometrického ověření, se kterými se můžeme setkat.

- Ověření obličeje - na základě analýzy rozpoznání obličejových rysů a vzorů.

- Ověření oční duhovky - založené na analýze snímku oční duhovky.

- Ověření oční sítnice - využívá se vzorů žil v zadní části oka.

- Ověření geometrie ruky - analýza geometrických prvků ruky, jako je délka a šířka jednotlivých prstů.

- Ověření hlasu - porovnání hlasového otisku.

- Ověření podpisu - na základě analýzy stylu rukopisu.

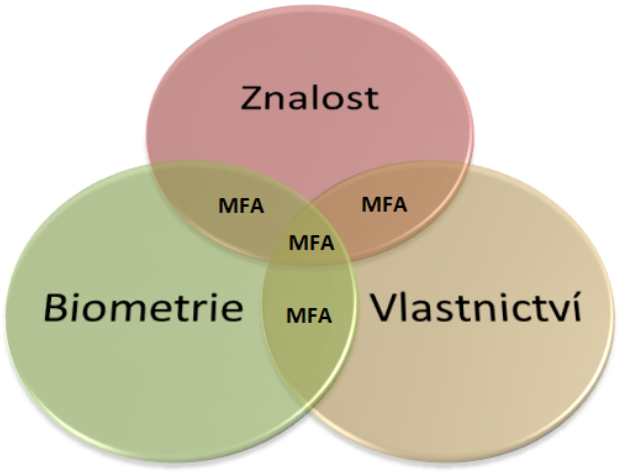
- A mnohé další ověření, jako je ověření na základě analýzy rychlosti psaní na klávesnici, tvaru ucha, chůze, pachu. [15]

Kapitola 3

Vícefaktorová autentizace

Metody z jednotlivých kategorií, jako je znalost, vlastnictví a biometrie mohou být kombinovány společně, abychom vytvořili vícefaktorovou autentizaci. Společná kombinace těchto faktorů je mnohem bezpečnější, než použití jednotlivých faktorů samostatně. Touto kombinací spojíme silné stránky jednotlivých faktorů a omezíme nebo zcela odstraníme jejich nedostatky. [2]

Vícefaktorová autentizace (MFA) je metoda ověřování, kdy pro udělení přístupu vyžadujeme dva nebo více autentizačních faktorů. Díky implementaci MFA můžeme snížit pravděpodobnost úspěšného kybernetického útoku a lépe tak ochránit své zařízení, síť, online účet a citlivé údaje. [16] Pokud by byl náhodou jeden z faktorů odcizen a zneužit, je malá šance, že bude zneužit i faktor druhý a to poskytuje bezpečnější způsob ověření identity uživatele. Vícefaktorovou autentizaci je dobré implementovat v závislosti na míře rizika. V situaci, kdy přistupujeme k citlivým údajům nebo provádíme transakce s větším objemem peněžních prostředků, potřebujeme silnější zabezpečení, než v případě, kdy se přihlašujeme k naši oblíbené online hře.



Obrázek 3.1: Vícefaktorová autentizace

U autentizace se můžeme setkat se zkratkami 2FA a MFA. Zkratka 2FA označuje dvoufaktorovou autentizace, která vždy využívá právě dva autentizační faktory pro ověření identity uživatele. Zkratka MFA označuje vícefaktorovou autentizaci a využívá dva nebo více autentizačních faktorů, využívá tedy libovolný počet faktorů, ale vždy více než jeden. [17]

Jako další typ vícefaktorové autentizace můžeme využít ověřování na základě polohy. Obvykle pomoci IP adresy zařízení nebo jeho geografické polohy. Pokud není IP adresa nebo daná poloha v seznamu povolených, můžeme uživateli zablokovat přístup. [16]

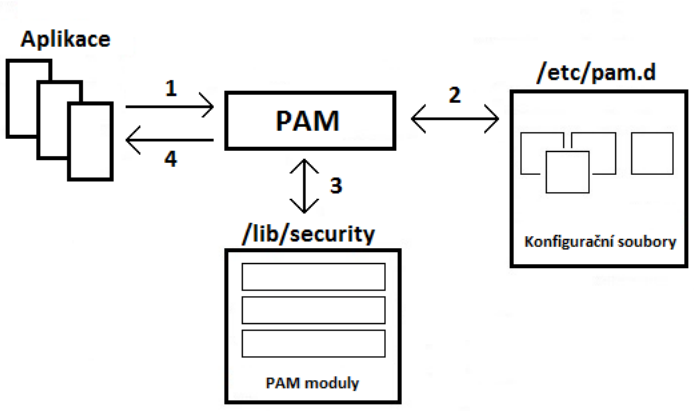
Také se můžeme setkát s pojmem adaptivní autentizace nebo autentizace založená na riziku, která zohledňuje kontext a chování uživatele při autentizaci pro přiřazení míry rizika při pokusu o autentizaci. Zohledňuje se například to, zda je připojení realizováno přes soukromou nebo veřejnou síť, zaměstnanec přistupuje k citlivým informacím během nebo mimo pracovní dobu nebo zda je použito stejné zařízení jako obvykle. Na základě těchto skutečností může být požadován další ověřovací faktor nebo je přístup zcela zamítnut. [16]

Mezi nejznámější případy MFA můžeme zařadit proces výběru peněž z bankomatu, kdy musíme použít svou platební kartu (vlastnictví) a zadat PIN (znalost). Další způsob využití MFA můžeme vidět u firem a korporací, kdy se zaměstnanci obvykle přihlašují ke svému pracovnímu učtu pomoci hesla a OTP kódu vygenerovaného v mobilní aplikaci. Nebo v případě přihlášení k online bankovnímu účtu, kdy je uživatel obvykle ověřen pomoci hesla a ověřovacího kódu, který mu přijde na mobilní telefon prostřednictvím textové SMS.

Kapitola 4

Autentizační PAM moduly v Linuxu

Pluggable Authentication Modules (PAM) je sada sdílených knihoven, které se využívají k dynamickému ověřování uživatelů. PAM je podporován mnoha moderními distribucemi Linuxu. PAM poskytuje robustnější autentizační prostředí, než by mohly poskytnout jednotlivé aplikace a služby. Díky zásuvné modulární architektuře poskytuje správci systému velkou flexibilitu při nastavení autentizace. Správce místního systému může pomoci PAM nakonfigurovat, jak se budou uživatelé ověřovat u konkrétních aplikací a služeb v systému Linux. Historicky měla každá aplikace svůj vlastní způsob autentizace uživatelů a musela být zkompilována pro použití tohoto konkrétního mechanismu autentizace. Jednotlivé moduly PAM obvykle najdeme v adresáři /lib/security, /lib64/security nebo /usr/lib/security v závislosti na architektuře a distribuci. V případě zařízení Raspberry Pi 4 a operačního systému Raspberry Pi OS (32-bit), které využívám v praktické částí této práce, jsou moduly umístěny v adresáři /lib/arm-linux-gnueabihf/security. Mezi hlavní benefity PAM patří určitě to, že je dobře zdokumentovaný a nabízí společné schéma autentizace pro široké spektrum aplikací. Vývojářům tedy umožňuje psát programy, aniž by museli vytvářet vlastní způsob autentizace. [18, 19]



Obrázek 4.1: Architektura PAM

V adresáři /etc/pam.d nalezneme konfigurační soubory pro jednotlivé aplikace a služby, které podporují PAM. Tyto konfigurační soubory definují, které moduly jsou volány, jejich pořadí a jak se má zacházet s výsledkem modulu. Každý konfigurační soubor v tomto adresáři má stejný název (malými písmeny) jako aplikace nebo služba, ke které řídí přístup. Dříve se prováděla konfigurace pomoci systémového souboru /etc/pam.conf, tento soubor se však nyní využívá pouze v případě, že neexistuje adresář /etc/pam.d. V případě existence adresáře /etc/pam.d je soubor /etc/pam.conf ignorován. Každá aplikace a služba, která využívá PAM je zodpovědná za definování názvu a instalaci svého konfiguračního souboru do adresáře /etc/pam.d. [18, 19]

```
pi@raspberrypi:/etc/pam.d $ ls
common-account                  lightdm             runuser
common-auth                     lightdm-autologin   runuser-l
common-password                 lightdm-greeter     sshd
common-session                  login               su
common-session-noninteractive   login.save          sudo
cron                            newusers            su-l
cups                            nginx               systemd-user
chfn                            other               vncserver
chpasswd                        passwd              xscreensaver
chsh                            polkit-1
pi@raspberrypi:/etc/pam.d $
```

Obrázek 4.2: Výpis adresaře /etc/pam.d

V konfiguračních souborech PAM je potřeba dodržovat určitou syntaxi, která je následující:

```
<Module type>     <Control flag>     <Module name>     <Module argument>
```

Kromě toho, že každá aplikace a služba, která podporuje PAM, by měla mít svůj vlastní konfigurační soubor v adresáři /etc/pam.d, je zde také soubor /etc/pam.d/other, kde můžeme nastavit zabezpečení pro služby, které nemají svůj vlastní konfigurační soubor. Je žádoucí, aby byl soubor /etc/pam.d/other dobře zabezpečený, protože tento soubor je výchozí konfigurací pro všechny aplikace podporující PAM a pokud má tento soubor slabiny, má slabiny i náš systém a může být zranitelnější vůči případnému útoku. Níže lze vidět paranoidní scénář v souboru /etc/pam.d/other, kdy je zakázáno vše, co není povoleno. [20]

```
auth       required   pam_deny.so
account    required   pam_deny.so
session    required   pam_deny.so
password   required   pam_deny.so
```

Listing 4.1: Příklad konfiguračního souboru /etc/pam.d/other

4.1 Module type

K dispozici jsou čtyři typy skupin pro správu.

- **auth** - Poskytuje autentizaci uživatele, tedy ověření, zda je uživatel opravdu tím, za koho se vydává. Například pomoci hesla, otisku prstu nebo hardwarového a softwarového tokenu.

- **account** - Provádí kontrolu uživatelského účtu, kontroluje, zda je uživateli povolen přístup k požadované službě, jestli nevypršela platnost hesla nebo zda má uživatel povolen přístup v danou dobu.

- **password** - Používá se k aktualizaci a změně uživatelských hesel.

- **session** - Provádí akce, které je pro uživatele potřeba vykonat před nebo po poskytnutí služby, jako je připojení domovského adresáře uživatele, uvítací hláška a podobně. [19]

4.2 Control flag

Všechny moduly PAM generují při volání výsledek jako úspěch 'pass' nebo selhání 'fail'. Kontrolní příznaky slouží k tomu, aby řekly PAM, co dělat s výsledkem modulu. Příznaky určují, jak je důležitý úspěch či selhání konkrétního modulu pro celkovou autentizaci k dané aplikaci či službě.

- **required** (povinný) - Pokud modul selže, bude uživatel upozorněn na selhání až po dokončení výsledků všech modulů v zásobníku pro tuto službu. Aby byl celkový výsledek autentizace úspěšný, musí být výsledek tohoto modulu úspěšný.

- **requisite** (bezpodmínečný) - Pokud modul selže, bude uživatel upozorněn na selhání daného modulu okamžitě a žádné další moduly se nevolají. Aby byl celkový výsledek autentizace úspěšný, musí být výsledek tohoto modulu úspěšný.

- **sufficient** (postačující) - Pokud takový modul uspěje a neselhal před ním žádný modul required, vrátí PAM výsledek autentizace jako úspěšný bez volání dalších modulů v zásobníku. Případné selhání modulu sufficient je ignorováno a volají se další moduly ze zásobníku.

- **optional** (nepovinný) - Výsledek tohoto modulu je důležitý pouze v případě, že se jedná o jediný modul v zásobníku u dané aplikace či služby, jinak je opomíjen.

- **include** (vložit) - Zahrnuje všechny řádky daného typu z konfiguračního souboru určeného jako argument tohoto ovládacího prvku. [19]

4.3 Module name

Jak již bylo zmíněno výše v této kapitole, jednotlivé PAM moduly se v operačním systému Raspberry Pi OS (32-bit) nacházejí v adresáři /lib/arm-linux-gnueabihf/security. Níže na obrázku 4.3 můžeme vidět výpis jednotlivých modulů umístěných v tomto adresáři. Většina těchto modulů je původních, ale některé autentizační moduly jsou zde přidány až mojí dodatečnou instalací, jsou to například moduly jako pam_u2f.so, pam_fprintd.so nebo pam_google_authenticator.so.

- **pam_unix.so** - Modul pro standardní ověření uživatele pomoci hesla. Hesla se ověřují proti souboru /etc/passwd a nebo /etc/shadow, kde jsou tato hesla uložena.
- **pam_time.so** - Tento modul neslouží pro ověření uživatele, ale může uživateli omezit přístup k aplikaci nebo službě v určitý časový interval, hodinu, den nebo dny v týdnu.
- **pam_deny.so** - Tento modul můžeme využít pro odepření přístupu, vždy indikuje selhání aplikace.
- **pam_permit.so** - Modul, který vždy povoluje přístup, může být velmi nebezpečný a měl by být používán s rozvahou. [18]
- **pam_google_authenticator.so** - Modul pro autentizaci uživatele pomoci softwarového tokenu (Google Authenticator).
- **pam_u2f.so** - Modul pro autentizaci uživatele pomoci hardwarového tokenu (YubiKey).
- **pam_fprintd.so** - Modul pro autentizaci uživatele pomoci otisku prstu.

```
pi@raspberrypi:/lib/arm-linux-gnueabihf/security $ ls
pam_access.so                pam_group.so       pam_nologin.so      pam_tally.so
pam_cifscreds.so             pam_chksshpwd.so   pam_permit.so       pam_tally2.so
pam_debug.so                 pam_issue.so       pam_pwhistory.so    pam_time.so
pam_deny.so                  pam_keyinit.so     pam_python.so       pam_timestamp.so
pam_echo.so                  pam_lastlog.so     pam_rhosts.so       pam_tty_audit.so
pam_env.so                   pam_limits.so      pam_rootok.so       pam_umask.so
pam_exec.so                  pam_listfile.so    pam_securetty.so    pam_unix.so
pam_faildelay.so             pam_localuser.so   pam_selinux.so      pam_userdb.so
pam_filter.so                pam_loginuid.so    pam_sepermit.so     pam_u2f.so
pam_fprintd.so               pam_mail.so        pam_shells.so       pam_warn.so
pam_ftp.so                   pam_mkhomedir.so   pam_stress.so       pam_wheel.so
pam_google_authenticator.la  pam_motd.so        pam_succeed_if.so   pam_xauth.so
pam_google_authenticator.so  pam_namespace.so   pam_systemd.so
pi@raspberrypi:/lib/arm-linux-gnueabihf/security $
```

Obrázek 4.3: PAM moduly

4.4 Module argument

Jakékoliv další pole, které se nachází za názvem modulu je argument nebo argumenty modulu. Každý modul má své vlastní argumenty. Argumenty modulu jsou odděleny mezerami a lze je využít k úpravě specifického chování daného modulu. [18]

- **audit** - Rozsáhlejší forma výpisu než debug.
- **debug** - Zapíše více informací do syslogu
- **nodelay** - Modul při selhání autentizace požaduje ve výchozím nastavení zpoždění , tento argument toto výchozí nastavení přepíše.
- **nullok** - Pokud je heslo prázdné ve výchozím nastavení autentizace selže, argument nullok toto původní nastavení přepíše.
- **use_first_pass** - Modul se pokusí použít heslo z předchozího autentizačního modulu, pokud je heslo nesprávné bude uživateli odepřen přístup. Nikdy nevyzve uživatele k zadání hesla.
- **try_first_pass** - Modul se pro ověření uživatele pokusí použít heslo z předchozího autentizačního modulu, pokud je toto heslo prázdné nebo nesprávné, bude uživatel vyzván k zadání nového hesla. [18]
- **cue** - Lze využít u modulu pam_u2f.so pro zobrazení výzvy uživateli.
- **interactive** - Zobrazí výzvu před ověřením přítomnosti zařízení YubiKey, vhodné pokud nemá zařízení hmatovou spoušť.

Kapitola 5

Embedded zařízení

Embedded zařízení je jednoúčelový počítač, který obsahuje speciální výpočetní systém. Systém je zcela zabudován do zařízení, které ovládá. Zařízení může nebo nemusí mít přístup k internetu. Taková zařízení se hojně využívají v průmyslu a zdravotnictví. Operační systém takového zařízení většinou spustí pouze jednu aplikaci, díky niž plní svou specifickou úlohu. Mezi embedded zařízení patří například myčky nádobí, bankomaty, směrovače, platební terminály, mikrovlnná trouba. Jelikož mají tyto zařízení omezený výpočetní výkon a přísné požadavky na napájení, vyžaduje psaní softwaru pro tyto zařízení znalost hardwarových komponent a programovaní. Díky tomu, že jsou tato zařízení většinou určena pro konkrétní účel, mohou být při návrhu zjednodušena a optimalizována pro hlavní aplikaci a tím dosáhneme snížení ceny tohoto zařízení. Další výhodou těchto zařízení je jejich jednoduchost a rychlost použití a také to, že mohou být vyráběna sériově ve velkém množství. Vzhledem k vlastnostem hardwaru můžeme za embedded zařízení označit také chytrý mobilní telefon, který je však z hlediska softwaru použitelný spíše jako osobní počítač. [21, 22]

Na trhu můžeme najít poměrně pestrou nabídku jednodeskových počítačů, kde mezi ty nejznámější patří bezesporu Raspberry Pi, které budu využívat také já v praktické části této diplomové práce. Konkrétně se jedná o zařízení Raspberry Pi 4 Comuter Model B (1 GB RAM). Toto zařízení je představeno níže v této kapitole, společně s dalšími alternativami, které se na trhu vyskytují a mohou být například levnější, výkonnější nebo téměř totožné. Nejde o to představit a vyjmenovat všechny možné alternativní zařízení, které se na trhu vyskytují, ale spíše pomoci několika zástupců ukázat, že neexistuje pouze Raspberry Pi.

5.1 RockPi 4 Model C

RockPi 4 Model C je vysoce výkonné zařízení, výbavou srovnatelné s Raspberry Pi 4. Je založeno na procesoru s architekturou ARM. Disponuje šesti-jádrovým procesorem Rockchip RK3399, až 4 GB LPDDR4 RAM a GPU Mali-T860 MP4. Obsahuje několik možností úložiště, včetně slotu

microSD a až 128 GB eMMC. K dispozici jsou dva video výstupy, jeden micro HDMI a druhý mini DisplayPort. Zařízení je napájeno prostřednictvím USB-C portu. [23]

- Micro HDMI až 4K při 60 Hz
- Mini DisplayPort 1.2 až 2560 x 1440 při 60 Hz
- Gigabit Ethernet s podporou PoE
- 802.11b/g/n/ac WiFi
- Bluetooth 5.0
- 1x USB 3.0, 1x USB 3.0 OTG, 2x USB 2.0
- 40-pin GPIO header s 1x UART, 2x SPI bus, 2x I2C bus a další standardní rozhraní

Obrázek 5.1: RockPi 4 Model C [24]

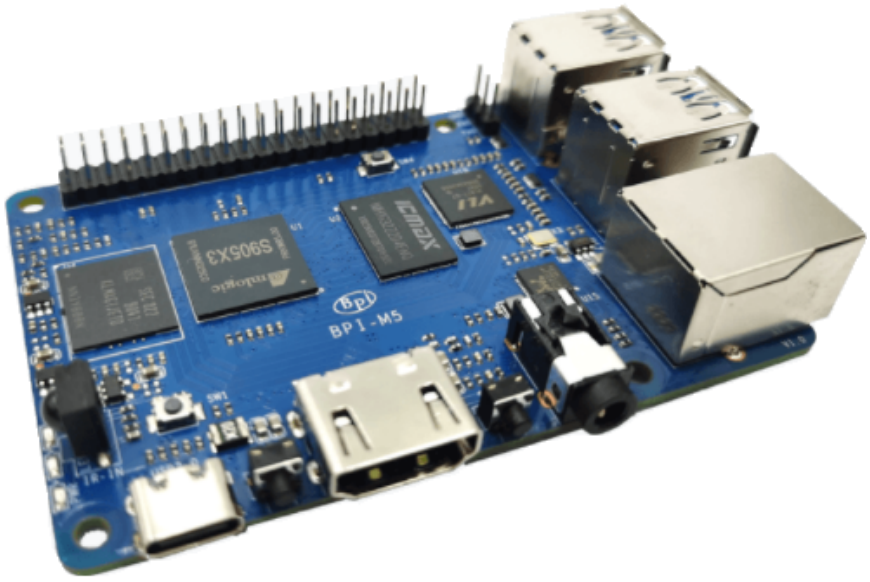
5.2 Onion Omega2Plus

Je malá vývojová deska založena na Linuxu. Vyniká hlavně cenou a malou velikostí. Díky nižšímu výkonu a velikosti je toto zařízení vhodné zejména pro jednodušší projekty a projekty IoT. Nabízí 580 MHz CPU, 128 MB RAM, 32 MB úložiště, USB 2.0 konektor, WiFi a slot pro microSD. V porovnání s Raspberry Pi nemá micro HDMI konektor, což může být značná nevýhoda. Pro rozšíření funkcionality je možné přikoupit několik doplňků. [25]

5.3 Banana Pi BPI-M5

Banana Pi BPI-M5 je jednodeskový počítač nové generace s podporou operačního systému Linux a Android. Využívá čtyř-jádrový procesor Amlogic S905X3 Cortex-A55, GPU Mali-G31 MP2, 4GB

LPDDR4 a 16 GB eMMC flash na desce. Má také microSD slot s podporou až 256 GB, 4 porty USB 3.0, 1x HDMI 2.0 až 4K při 60 Hz, Gigabit Ethernet, standardní 40 pinový GPIO. [26]



Obrázek 5.2: Banana Pi BPI-M5 [26]

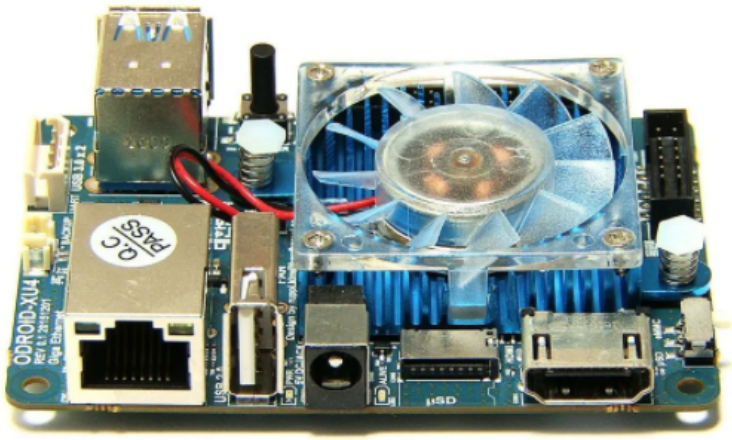
5.4 PocketBeagle

PocketBeagle je ultratenký, jednodeskový open-source počítač o velikosti 56 mm x 35 mm, přibližně stejně velký jako Raspberry Pi Zero. Vyznačuje se nízkou cenou a jednoduchým použitím. Hodí se pro začátečníky i profesionály. Můžeme ho naprogramovat pomocí webového prohlížeče, který poskytuje přístup k příkazovému řádku a textovému editoru Linuxu. Je postaven na procesoru Octavo Systems OSD3358 1 GHz ARM Cortex-A8 a integrované 512 MB DDR3 RAM. Obsahuje konektor microUSB, slot microSD a 72 rozšiřujících pinů. [25]

Obrázek 5.3: PocketBeagle [25]

5.5 Odroid-XU4

ODROID-XU4 je jednodeskový počítač menšího formátu s výkonnějším a energeticky efektivnějším hardwarem. Má zabudované aktivní chlazení pomoci ventilátoru. Obsahuje procesor Samsung Exynos 5422 (4x Cortex-A15 @ 2,0 GHz a 4x Cortex-A7 @ 1,4 GHz), GPU Mali-T628 MP6 a 2 GB RAM. Podporuje různé verze systému Linux, včetně verzí Ubuntu a Android. Má implementované eMMC 5.0, USB 3.0 a Gigabit Ethernet, díky tomu poskytuje vysokou rychlost přenosu dat, což uživatele pocítí zejména u rychlejšího spuštění, procházení webu a dokonce i hraní 3D her. Chybí WiFi a Bluetooth a proto pro připojení k internetu musíme využít WiFi modul připojený přes USB nebo ethernetový kabel. [25]



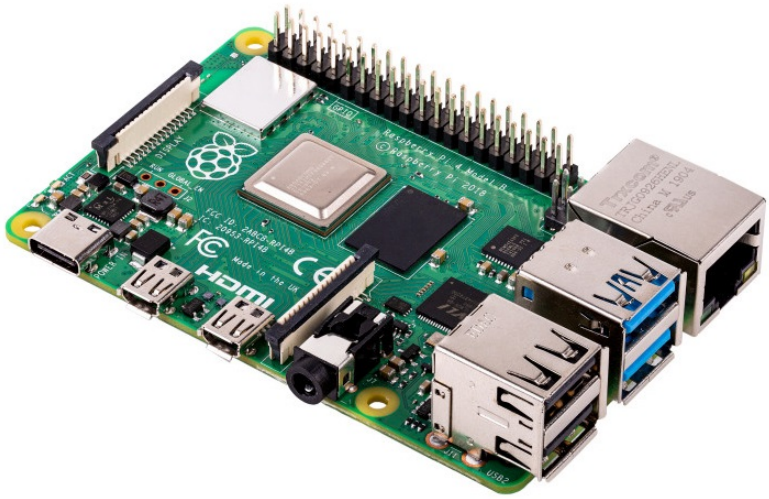
Obrázek 5.4: Odroid-XU4 [25]

5.6 Raspberry Pi 4 Computer Model B

Raspberry Pi patří mezi nejoblíbenější jednodeskové počítače. Verze 4 Model B představuje zatím nejvýkonnější model od tohoto výrobce. Tento model můžeme považovat za plnohodnotnou náhradu klasického počítače. Raspberry Pi Model B se drží stejného vzhledu jako jeho předchůdci, došlo ovšem k několika změnám v oblasti konektorů. Díky těmto změnám je například nutné pořídit novou krabičku. Obsahuje napájecí konektor USB-C a dva micro HDMI konektory, díky nimž lze připojit až dva monitory. Raspberry Pi nabízí modely s paměti typu LPDDR4 o velikostech 1 GB RAM, 2 GB RAM nebo 4GB RAM. Podpora operačního systému Linux a Android.

- čtyřjádrový procesor ARM Cortex-A72
- 2x USB 3.0, 2x USB 2.0
- 2x micro HDMI (podpora 4Kp60)

- Gigabit Ethernet
- 802.11b/g/n/ac WiFi
- Bluetooth 5.0
- schopný PoE
- standardní GPIO header se 40 piny
- Micro SD slot pro operační systém a ukládání dat [27]



Obrázek 5.5: Raspberry Pi 4 Computer Model B [27]

Kapitola 6

Vícefaktorová autentizace pro fyzický přístup k zařízení Raspberry Pi

6.1 Návrh MFA pro přihlášení a služby - varianta a

Návrh MFA pro přihlášení a služby, kdy zařízení Raspberry Pi využíváme jako klasický počítač a máme k němu připojeny všechny periferní zařízení. Pro ověření uživatele v systému budeme v této variantě využívat všech tří autentizačních faktorů:

```
Heslo (znalost)
Statické heslo na YubiKey 5 NFC (vlastnictví)
Otisk prstu (biometrie)
```

Jako první si zaregistrujeme otisk prstu uživatele, pomoci čtečky otisku prstu Upek eikon a následně si pomoci nástroje *YubiKey Personalization Tool* nastavíme statické heslo na YubiKey 5 NFC, které pak využijeme při vytvoření nového uživatelského hesla. Toto nové heslo se bude skládat z první částí, kterou si uživatel bude pamatovat a druhé části, kterou si uživatel pamatovat nebude a ta bude generována jako statické heslo z YubiKey 5 NFC.

6.1.1 Čtečka otisků prstů Upek eikon

Jako zástupce biometrické autentizace budu využívat čtečku otisku prstu Upek eikon. Čtečka se k Raspberry Pi připojuje pomoci rozhraní USB. Čtečka působí bytelně a dobře sedí na pracovní ploše, což přispívá k pohodlnějšímu získání otisku prstu. Podle výrobce má tato čtečka podporu v operačních systémech Windows, Linux a macOS. Typ senzoru čtečky je kapacitní CMOS senzor. Biometrickou autentizaci pomoci čtečky otisku prstu je vhodnější implementovat v případě, kdy Raspberry Pi využíváme jako klasický počítač a máme k němu připojeny všechny periferní zařízení, včetně čtečky otisku prstu Upek eikon, než při vzdáleném přístupu přes protokol SSH, kdy využíváme jiné zařízení, které čtečku otisku prstu mít nemusí.

Obrázek 6.1: Upek eikon [28]

Jako první připojíme čtečku otisku prstu Upek eikon do volného USB konektoru na Raspberry Pi a ujistíme se, zda je zařízení aktivní. To provedeme pomoci příkazu *lsusb* v terminálu. Na obrázku 6.2 můžeme vidět, že zařízení bylo identifikováno a je aktivní, vše je tedy v pořádku.

```
pi@raspberrypi:~ $ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 004: ID 040b:2000 Weltrend Semiconductor
Bus 001 Device 005: ID 1bcf:0005 Sunplus Innovation Technology Inc. Optical Mouse
Bus 001 Device 003: ID 147e:2016 Upek Biometric Touchchip/Touchstrip Fingerprint Sensor
Bus 001 Device 006: ID 1050:0407 Yubico.com Yubikey 4 OTP+U2F+CCID
Bus 001 Device 002: ID 2109:3431 VIA Labs, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
pi@raspberrypi:~ $
```

Obrázek 6.2: Kontrola připojení čtečky Upek eikon

Nainstalujeme potřebné balíčky.

```
pi@raspberrypi:~ $ sudo apt install -y fprintd libpam-fprintd
```

Listing 6.1: Instalace balíčků pro Upek eikon

Nyní zaregistrujeme otisk prstu uživatele pomoci příkazu *fprintd-enroll [uživatelské jméno]*. V našem případě se jedná o uživatele *pi*. Postup registrace můžeme vidět na obrázku 6.3. Jsme vyzváni, abychom několikrát za sebou přiložili pravý ukazováček, dokud není registrace kompletní. Pokud nedojde při registrace k žádné chybě, má uživatel *pi* úspěšně zaregistrován svůj otisk prstu.

Dále je možné využít příkazy *fprintd-delete [uživatelské jméno]* pro odstranění otisku prstu daného uživatele nebo příkaz *fprintd-list [uživatelské jméno]* pro zobrazení seznamu zaregistrovaných otisků prstů uživatele. Před tím, než se rozhodneme implementovat autentizaci pomoci otisku prstu do našeho systému, je dobré si otisk prstu nejprve ověřit pomoci příkazu *fprintd-verify [uživatelské jméno]*. Na obrázku 6.4 můžeme vidět toto úspěšné ověření.

```
pi@raspberrypi:~ $ fprintd-enroll pi
Using device /net/reactivated/Fprint/Device/0
Enrolling right-index-finger finger.
Enroll result: enroll-stage-passed
Enroll result: enroll-stage-passed
Enroll result: enroll-stage-passed
Enroll result: enroll-retry-scan
Enroll result: enroll-stage-passed
Enroll result: enroll-completed
pi@raspberrypi:~ $
```

Obrázek 6.3: Registrace otisku prstu uživatele pi

```
pi@raspberrypi:~ $ fprintd-verify pi
Using device /net/reactivated/Fprint/Device/0
Listing enrolled fingers:
 - #0: right-index-finger
Verify result: verify-match (done)
pi@raspberrypi:~ $
```

Obrázek 6.4: Úspěšné ověření otisku prstu

Na obrázku 6.5 můžeme pro ukázku vidět neúspěšné ověření. Při prvním pokusu o ověření nebyl prst snímán dostatečně dlouhou dobu, jednalo se totiž jen o velmi krátké přiložení prstu a při dalším pokusu byl přiložen nesprávný prst.

```
pi@raspberrypi:~ $ fprintd-verify pi
Using device /net/reactivated/Fprint/Device/0
Listing enrolled fingers:
 - #0: right-index-finger
Verify result: verify-swipe-too-short (not done)
Verify result: verify-no-match (done)
pi@raspberrypi:~ $
```

Obrázek 6.5: Neúspěšné ověření otisku prstu

Nyní máme zaregistrován a ověřen otisk prstu pro uživatele *pi* a můžeme pomoci autentizačního modulu *pam_fprintd.so* nastavit ověření uživatele pomoci otisku prstu pro aplikace a služby.

6.1.2 YubiKey 5 NFC

Abychom mohli plně využít možností YubiKey 5 NFC, je potřeba nainstalovat některé balíčky, pokud nejsou tyto balíčky pro YubiKey v naši distribuci OS k dispozici, využijeme možnost instalace pomoci PPA Yubico.

```
pi@raspberrypi:~ $ sudo add-apt-repository ppa:yubico/stable
pi@raspberrypi:~ $ sudo apt-get update
```

Listing 6.2: PPA Yubico

Nyní můžeme instalovat nejnovější software Yubico.

6.1.2.1 YubiKey Personalization Tool

Pomoci softwaru *YubiKey Personalization Tool* můžeme zjistit informace o našem YubiKey, jeho sériové číslo, verzi, podporované protokoly a také provádět jeho konfiguraci. YubiKey 5 NFC má k dispozici dva sloty pro konfiguraci. Tyto sloty můžeme využít pro uložení statického hesla nebo protokoly jako je OTP, OATH-HOTP. Jednotlivé sloty lze po konfiguraci aktivovat tak, že při krátkém přidržení prstu (přibližně sekundu) na kovovém kontaktu YubiKey se generují data ze slotu 1 a při delším přidržení prstu (přibližně 3 sekundy) se generují data ze slotu 2.

Příkaz v terminálu pro instalaci softwaru *YubiKey Personalization Tool*.

```
pi@raspberrypi:~ $ sudo apt install yubikey-personalization-gui
```

Listing 6.3: Instalace YubiKey Personalization Tool

6.1.2.2 Nastavení statického hesla pomoci YubiKey Personalization Tool

Spuštění grafického prostředí *YubiKey Personalization Tool* provedeme pomoci příkazu

```
pi@raspberrypi:~ $ yubikey-personalization-gui
```

Listing 6.4: Spuštění YubiKey Personalization Tool

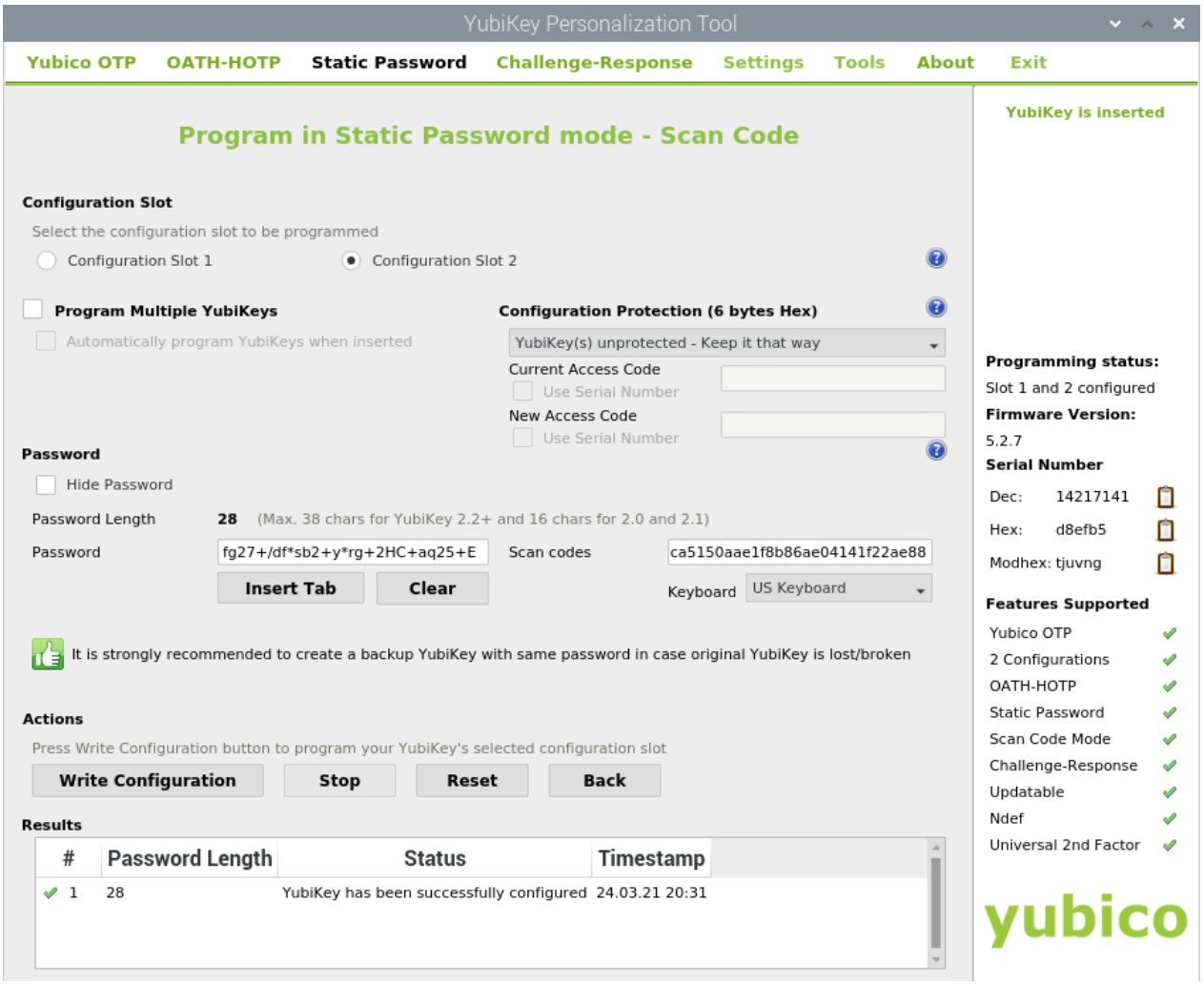
nebo v levém horním rohu klikneme na Malinu -> Příslušenství -> YubiKey Personalization Tool. Po spuštění zvolíme možnost *Static Password* a dále možnost *Scan Code*. Na obrázku 6.6 můžeme vidět, jak vypadá nastavení statického heslo v *YubiKey Personalization Tool*. Nejprve si vybereme konfigurační slot, na který chceme statické heslo uložit. Nalevo si můžeme všimnout, že oba tyto sloty jsou už zabrány, protože jsem už dříve statické heslo na slot 2 konfiguroval, původně byl tento slot volný, to však nevadí a slot 2 si jednoduše přepíšeme novým statickým heslem. Do pole *Password* zadáme naše dlouhé, těžko zapamatovatelné heslo a zapíšeme ho na náš YubiKey 5 NFC pomoci možnosti *Write Configuration*.

Jak už bylo zmíněno výše, statické heslo na YubiKey 5 NFC budeme využívat jako druhou, obtížně zapamatovatelnou část hesla, k první části hesla, kterou si pamatovat budeme. Tímto způsobem výrazně zvýšíme náročnost prolomení našeho hesla.

Budeme mít například heslo:

```
Heslo12Heslolafg27+/df*sb2+y*rg+2HC+aq25+E
```

Část hesla 'Heslo12Heslo' si budeme pamatovat a druhou část 'fg27+/df*sb2+y*rg+2HC+aq25+E' budeme generovat pomoci YubiKey. Je dobré využívat statické heslo tímto způsobem, kdy i nějakou část hesla známe, kdybychom totiž generovali celé heslo pomoci YubiKey, stačilo by, aby se někdo zmocnit našeho YubiKey a měl by jednoduše otevřen přístup do našeho systému, účtu či aplikace. Touto kombinací využíváme dvou faktorů a to znalosti a vlastnictví. V případě generování celého hesla pomoci YubiKey bychom využívali pouze jednoho faktoru.



Obrázek 6.6: YubiKey Personalization Tool

6.1.3 Nastavení nového uživatelského hesla

Změna hesla se provede pomoci následujícího příkazu:

```
pi@raspberrypi:~ $ passwd
```

Listing 6.5: Příkaz pro změnu hesla

Nejprve jsme požádáni o stávající heslo a poté můžeme zadat heslo nové. To provedeme tak, že nejprve napíšeme část hesla, kterou si budeme pamatovat a nedáme *Enter*, ale přiložíme prst na YubiKey 5 NFC po dobu přibližně 3 sekund. YubiKey vygeneruje statického hesla ze slotu 2 a sám provede potvrzení. Stejný postup zopakujeme při potvrzení hesla.

6.1.4 Úprava konfiguračního souboru PAM

Otevřeme si konfigurační soubor /etc/pam.d/common-auth, kde se provádí nastavení autentizace společně pro všechny služby.

```
pi@raspberrypi:~ $ sudo nano /etc/pam.d/common-auth
```

Listing 6.6: Otevření konfiguračního souboru v editoru nano

Autentizace pomoci hesla je zde standardně implementována pomoci modulu pam_unix.so.

```
auth    [success=1 default=ignore]   pam_unix.so  nullok_secure
```

Listing 6.7: Ověření pomoci hesla

Zbývá nám tedy do tohoto konfiguračního souboru přidat autentizaci pomoci čtečky otisku prstu Upek eikon, to provedeme pomoci následujícího řádku, který přidáme nejlépe na konec tohoto konfiguračního souboru a soubor uložíme.

```
auth    required   pam_fprintd.so
```

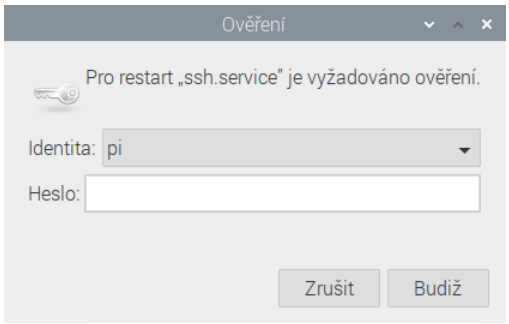
Listing 6.8: Ověření pomoci otisku prstu

6.1.5 Ověření vícefaktorové autentizace

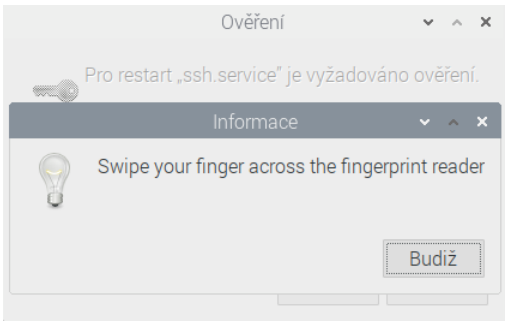
Vícefaktorovou autentizaci pro přihlášení a služby máme implementovánu, nyní ověříme, zda vše v pořádku funguje. MFA si můžeme ověřit při přihlášení uživatele nebo třeba při restartu služby SSH. Zadáme příkaz pro restart služby SSH.

```
pi@raspberrypi:~ $ service ssh restart
```

Listing 6.9: Restart SSH



(a) Ověření pomoci hesla



(b) Ověření pomoci otisku prstu

Obrázek 6.7: Autentizace uživatele pro restart služby SSH - varianta a

Na obrázku 6.7 lze vidět, že se nejprve objeví okno pro ověření uživatele pomoci hesla, to je způsobeno tím, že je modul pro ověření uživatele pomoci hesla pam_unix.so v konfiguračním souboru výše, než modul pro ověření pomoci otisku prstu. Zadáme tedy nám známou část hesla a poté podržíme prst na kovovém kontaktu YubiKey 5 NFC (3 sekundy), pro vygenerování statického hesla. Po úspěšném ověření hesla přijde na řadu ověření pomoci čtečky otisku prstu. Pokud je úspěšně ověřen i otisk prstu, autentizace je úspěšná a dojde k restartu služby SSH. Ze zkušenosti s používáním čtečky otisku prstu Upek eikon musím však přiznat, že úspěšné ověření otisku prstu nemusí být vždy snadné, protože relativně často dochází k chybnému ověření i při správném procesu použití čtečky. U generování statického hesla z YubiKey 5 NFC nebyly shledány žádné problémy.

Na obrázku 6.8 můžeme vidět výpis souboru /var/log/auth.log při úspěšné autentizaci, kde můžeme pozorovat proces autentizace pomoci jednotlivých autentizačních metod a následný restart služby SSH.

```
Apr 11 15:25:00 raspberrypi polkitd(authority=local): Registered Authentication Agent for unix-process:3152:470519 (system bus name :1.224
[/usr/bin/pkttyagent --notify-fd 5 --fallback], object path /org/freedesktop/PolicyKit1/AuthenticationAgent, locale cs_CZ.UTF-8)
Apr 11 15:25:07 raspberrypi polkit-1[3161]: pam_fprintd(polkit-1): Using device /net/reactivated/Fprint/Device/0
Apr 11 15:25:07 raspberrypi polkit-1[3161]: pam_fprintd(polkit-1): prints registered: yes
Apr 11 15:25:07 raspberrypi polkit-1[3161]: pam_fprintd(polkit-1): verify_finger_selected Swipe your finger across the fingerprint reader
Apr 11 15:25:10 raspberrypi polkit-1[3161]: pam_fprintd(polkit-1): Verify result: verify-match
Apr 11 15:25:14 raspberrypi polkitd(authority=local): Operator of unix-session:14 successfully authenticated as unix-user:pi to gain TEMPOR
ARY authorization for action org.freedesktop.systemd1.manage-units for system-bus-name::1.225 [systemctl restart ssh.service] (owned by uni
x-user:pi)
Apr 11 15:25:14 raspberrypi sshd[3037]: Received signal 15; terminating.
Apr 11 15:25:14 raspberrypi sshd[3171]: Server listening on 0.0.0.0 port 22.
Apr 11 15:25:14 raspberrypi sshd[3171]: Server listening on :: port 22.
Apr 11 15:25:14 raspberrypi polkitd(authority=local): Unregistered Authentication Agent for unix-process:3152:470519 (system bus name :1.22
4, object path /org/freedesktop/PolicyKit1/AuthenticationAgent, locale cs_CZ.UTF-8) (disconnected from bus)
```

Obrázek 6.8: Výpis souboru /var/log/auth.log - varianta a

Na obrázku 6.9 lze vidět výpis souboru /var/log/syslog při úspěšné autentizaci.

```
Apr 11 15:25:07 raspberrypi dbus-daemon[369]: [system] Activating via systemd: service name='net.reactivated.Fprint' unit='fprintd.service'
 requested by ':1.227' (uid=0 pid=3161 comm="/usr/lib/policykit-1/polkit-agent-helper-1 pi ")
Apr 11 15:25:07 raspberrypi systemd[1]: Starting Fingerprint Authentication Daemon...
Apr 11 15:25:07 raspberrypi dbus-daemon[369]: [system] Successfully activated service 'net.reactivated.Fprint'
Apr 11 15:25:07 raspberrypi systemd[1]: Started Fingerprint Authentication Daemon.
Apr 11 15:25:14 raspberrypi systemd[1]: Stopping OpenBSD Secure Shell server...
Apr 11 15:25:14 raspberrypi systemd[1]: ssh.service: Succeeded.
Apr 11 15:25:14 raspberrypi systemd[1]: Stopped OpenBSD Secure Shell server.
Apr 11 15:25:14 raspberrypi systemd[1]: Starting OpenBSD Secure Shell server...
Apr 11 15:25:14 raspberrypi systemd[1]: Started OpenBSD Secure Shell server.
Apr 11 15:25:40 raspberrypi systemd[1]: fprintd.service: Succeeded.
```

Obrázek 6.9: Výpis souboru /var/log/syslog - varianta a

6.2 Návrh MFA pro přihlášení a služby - varianta b

Druhý návrh MFA pro přihlášení a služby, kdy Raspberry Pi využíváme jako klasický stolní počítač a nepřistupujeme k němu pouze pomoci vzdálené konzole. Tato varianta MFA se bude skládat ze dvou autentizačních faktorů:

`Heslo (znalost)`
`YubiKey 5 NFC - Universal 2nd Factor (vlastnictví)`

6.2.1 YubiKey 5 NFC - Universal 2nd Factor

Následná konfigurace složí pro autentizaci pomoci standardu U2F v rámci místního systému, jako je přihlášení, sudo, služby a aplikace. Nefunguje však pro autentizaci u SSH.

Instalace potřebných balíčků.

```
pi@raspberrypi:~ $ sudo apt-get install libpam-u2f
```

Listing 6.10: Instalace balíčku libpam-u2f

Zařízení YubiKey 5 NFC připojíme pomoci USB konektoru k zařízení Raspberry Pi a v terminálu zadáme následující příkazy pro vytvoření adresáře a přiřazení U2F klíče k našemu účtu.

```
pi@raspberrypi:~ $ mkdir -p ~/.config/Yubico
pi@raspberrypi:~ $ pamu2fcfg > ~/.config/Yubico/u2f_keys
```

Listing 6.11: Nastavení YubiKey (U2F)

V tomto okamžiku začne zařízení YubiKey 5 NFC blikat, přiložíme tedy prst na kovový kontakt YubiKey 5 NFC pro potvrzení a to je vše. Pokud chceme zvýšit zabezpečení, můžeme přesunout soubor u2f_keys do oblasti operačního systému, kde je vyžadováno sudo k úpravám souboru. V případě tohoto přesunu je však potřeba upravit modul pam_u2f.so, který se nachází v adresáři /lib/arm-linux-gnueabihf/security. Úprava spočívá v tom, že na konec tohoto modulu přidáme řádek, který definuje cestu k novému umístění souboru například takto:

```
authfile=/etc/Yubico/u2f_keys
```

Listing 6.12: Definování cesty k souboru u2f_keys v modulu pam_u2f.so

Nyní můžeme využívat modul pam_u2f.so pro implementaci autentizace pomoci U2F v zařízení Raspberry Pi.

6.2.2 Úprava konfiguračního souboru PAM

Otevřeme si opět konfigurační soubor /etc/pam.d/common-auth, pro nastavení společné autentizace pro všechny služby.

```
pi@raspberrypi:~ $ sudo nano /etc/pam.d/common-auth
```

Listing 6.13: Otevření konfiguračního souboru v editoru nano

Jak už víme, autentizace pomoci hesla je zde standardně implementována modulem pam_unix.so.

```
auth    [success=1 default=ignore]   pam_unix.so nullok_secure
```

Listing 6.14: Ověření pomoci hesla

Do konfiguračního souboru tedy stačí přidat autentizaci pomoci modulu pam_u2f.so. To provedeme následujícím řádkem, který přidáme na konec tohoto konfiguračního souboru. Jelikož tento modul přidáme na konec konfiguračního souboru, bude nejprve uživatel ověřen pomoci hesla a až poté pomoci YubiKey, moduly se totiž volají podle pořadí v konfiguračním souboru. Můžeme si všimnout argumentu *cue*, který slouží pro zobrazení informační hlášky uživateli při autentizaci a *debug* pro povolení výpisu.

```
auth    required    pam_u2f.so  debug cue
```

Listing 6.15: Autentizace pomoci modulu pam_u2f.so (YubiKey)

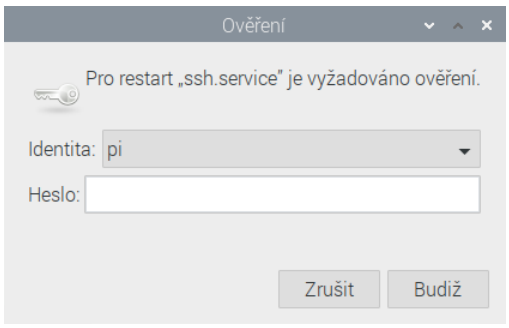
6.2.3 Ověření vícefaktorové autentizace

Vícefaktorovou autentizace opět ověříme u restartu služby SSH.

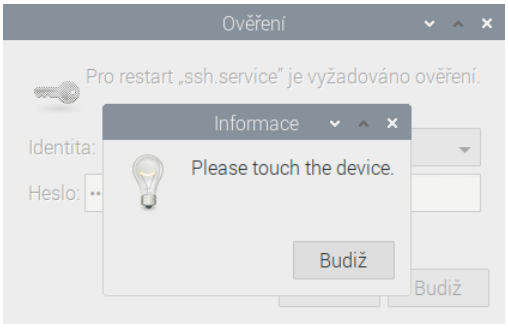
```
pi@raspberrypi:~ $ service ssh restart
```

Listing 6.16: Příkaz pro restart služby SSH

Průběh autentizace lze vidět na obrázku 6.10, uživatel je nejprve ověřen heslem a poté pomoci YubiKey, kdy je požádán, aby se dotkl tohoto zařízení. Informační zpráva 'Please touch the device' se zobrazí díky argumentu *cue*, bez tohoto argumentu by se žádná informační zpráva nezobrazila a zařízení YubiKey 5 NFC by bez oznámení začalo blikat a čekalo na dotek uživatele.



(a) Ověření pomoci hesla



(b) Ověření pomoci YubiKey

Obrázek 6.10: Autentizace uživatele pro restart služby SSH - varianta b

Na obrázku 6.11 můžeme vidět výpis ze souboru /var/log/auth.log při úspěšné autentizaci, můžeme si všimnout, že zde není žádná informace o autentizaci pomoci YubiKey 5 NFC. Varianta této MFA pomoci hesla a YubiKey 5 NFC se osvědčila velmi dobře a nevyskytly se zde žádné problémy při ověření uživatele, jako třeba u čtečky otisku prstu Upek eikon.

```
Apr 13 14:41:52 raspberrypi polkitd(authority=local): Registered Authentication Agent for unix-process:1281:343494 (system bus name :1.44
 [/usr/bin/pkttyagent --notify-fd 5 --fallback], object path /org/freedesktop/PolicyKit1/AuthenticationAgent, locale cs_CZ.UTF-8)
Apr 13 14:42:04 raspberrypi polkitd(authority=local): Operator of unix-session:1 successfully authenticated as unix-user:pi to gain TEMPO
RARY authorization for action org.freedesktop.systemd1.manage-units for system-bus-name::1.45 [systemctl restart ssh.service] (owned by u
nix-user:pi)
Apr 13 14:42:04 raspberrypi sshd[1270]: Received signal 15; terminating.
Apr 13 14:42:04 raspberrypi sshd[1298]: Server listening on 0.0.0.0 port 22.
Apr 13 14:42:04 raspberrypi sshd[1298]: Server listening on :: port 22.
Apr 13 14:42:04 raspberrypi polkitd(authority=local): Unregistered Authentication Agent for unix-process:1281:343494 (system bus name :1.
44, object path /org/freedesktop/PolicyKit1/AuthenticationAgent, locale cs_CZ.UTF-8) (disconnected from bus)
```

Obrázek 6.11: Výpis souboru /var/log/auth.log - varianta b

6.3 Návrh MFA pro přihlášení a služby - varianta c

Třetí návrh MFA pro přihlášení a služby, kdy zařízení Raspberry Pi využíváme jako klasický stolní počítač, ke kterému nepřistupujeme pouze přes SSH. V této variantě MFA budeme využívat dva autentizační faktory:

`RFID přístupová karta 125kHz (vlastnictví)`
`Otisk prstu (biometrie)`

6.3.1 Čtečka kódů RFID

Pro autentizaci budu využívat RFID čtečku a RFID kartu pracující na frekvence 125 kHz.

Obrázek 6.12: RFID čtečka [29]

Tato čtečka RFID karet/čipů je poměrně jednoduché zařízení, které po připojení k počítači automaticky nainstaluje driver a funguje jako emulace klávesnice. Po přiložení karty ke čtečce, čtečka vždy vypíše číslo karty. Čtečku při autentizaci využijeme tak, že nastavíme uživatelské heslo jako číslo naši RFID karty, kterou použijeme při autentizaci. Budeme tedy opět pro ověření využívat modul pam_unix.so. Nastavení nového uživatelského hesla:

```
pi@raspberrypi:~ $ passwd
```

```
pi@raspberrypi:~ $ passwd
Změna hesla pro pi.
Current password:
Nové heslo:
Opakujte nové heslo:
passwd: heslo bylo úspěšně změněno
pi@raspberrypi:~ $
```

Obrázek 6.13: Změna hesla

Nejprve zadáme původní heslo a při zadávání nového hesla přiložíme naši RFID kartu k RFID čtečce, přiložení zopakujeme pro potvrzení hesla, nyní je nastaveno heslo na číslo naši RFID karty.

6.3.2 Úprava konfiguračního souboru PAM

Otevřeme si opět konfigurační soubor /etc/pam.d/common-auth v textovém editoru nano.

```
pi@raspberrypi:~ $ sudo nano /etc/pam.d/common-auth
```

Listing 6.17: Otevření konfiguračního souboru v editoru nano

Jak už jsem zmínil výše, pro autentizaci pomoci naši RFID karty, budeme opět využívat standardní modul pam_unix.so, pro ověření pomoci hesla.

```
auth    [success=1 default=ignore]   pam_unix.so nullok_secure
```

Listing 6.18: Ověření pomoci RFID karty

Přidání autentizace pomoci čtečky otisku prstu Upek eikon jsme si už jednou ukazovali a provedeme to opět pomoci následujícího řádku, který přidáme na konec tohoto konfiguračního souboru a soubor uložíme.

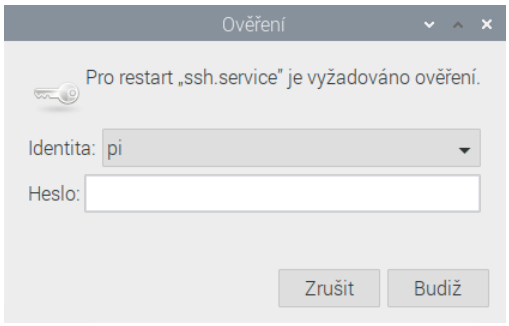
```
auth    required   pam_fprintd.so
```

Listing 6.19: Ověření pomoci otisku prstu

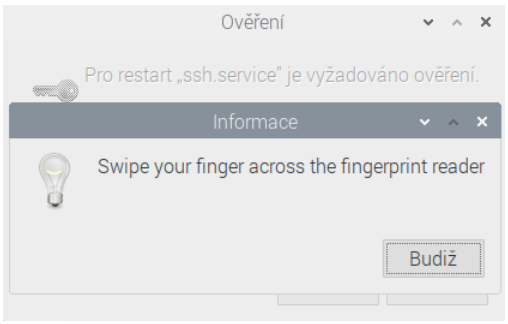
6.3.3 Ověření vícefaktorové autentizace

Vícefaktorovou autentizaci si znovu ověříme při restartu služby SSH. Na obrázku 6.14 můžeme vidět, že při autentizaci je nejprve vyžadováno uživatelské heslo, kdy pro zadání tohoto hesla využijeme naši RFID kartu, kterou přiložíme k RFID čtečce. Ve druhém kroku autentizace jsme ověřeni pomoci otisku prstu. Po větším otestování této varianty MFA, kdy využíváme faktoru vlastnictví (RFID karta) a biometrie (otisk prstu), se zdá tato varianta MFA jako uživatelský velmi přívětivá. U ověření pomoci RFID karty se nevyskytnul žádný problém a skvěle spolupracuje s modulem pam_unix.so. Hlavní nevýhodou této RFID čtečky je však skutečnost, že při přiložení RFID karty k RFID čtečce

se vždy vypíše číslo RFID karty, tedy i v případě, kdy nedochází k autentizaci a to může být potenciálně nebezpečné.



(a) Ověření pomoci RFID karty



(b) Ověření pomoci otisku prstu

Obrázek 6.14: Autentizace uživatele pro restart služby SSH - varianta c

Ve výpisech v souboru /var/log/auth.log a /var/log/syslog nejsou žádné změny a jsou totožné s výpisy u varianty a, protože při této variantě MFA využíváme stejné autentizační moduly.

6.4 Další možnosti a varianty vícefaktorové autentizace

6.4.1 Google Authenticator

Pro nastavení této metody autentizace využijeme kromě Raspberry Pi také náš chytrý mobilní telefon (iOS nebo Android), kde si nainstalujeme aplikaci OATH-TOTP jako je Google Authenticator, která generuje jednorázové hesla. Obvykle se jedná o šestimístné číslo, které se mění každých 30 sekund. Google kromě této aplikace vytvořil také modul PAM, který je plně kompatibilní s touto aplikaci.

```
pi@raspberrypi:~ $ sudo apt-get install libpam-google-authenticator
pi@raspberrypi:~ $ google-authenticator
```

Listing 6.20: Instalace a spuštění google_authenticator na Raspberry Pi

Nainstalujeme potřebný baliček a spustíme inicializační aplikaci *google-authenticator*, po spuštění jsme dotázani, zda mají být autentizační tokeny založeny na čase? Zvolíme *ano*, protože to aplikace Google Authenticator vyžaduje. Po zodpovězení tohoto dotazu se vygeneruje QR kód.

Tento QR kód naskenujeme pomoci aplikace Google Authenticator v mobilním telefonu. Po naskenování se nám v aplikaci zobrazí šestimístný kód, který se mění každých 30 sekund. Než budeme v Raspberry Pi pokračovat, nezapomeňme si poznamenat kódy pro obnovení, protože je to jediný způsob, jak případně obnovit přístup. Následují další dotazy ohledně fungovaní modulu PAM. Na všechny dotazy můžeme odpovědět *ano*, jen u dotazu, zda chceme zvětšit okno pro platnost tokenu, můžeme pro lepší zabezpečení odpovědět *ne*.

Po úspěšné konfiguraci, můžeme začít využívat také modul pam_google_authenticator.so pro vícefaktorové autentizace. Je důležité zmínit, že autentizaci pomoci Google Authenticator nyní můžeme využívat pouze uživatel pi, který byl přihlášen při této konfiguraci. Každý další uživatel, který chce využívat tento způsob autentizace, se bude muset přihlásit a spustit sám inicializační aplikaci, aby získal svůj vlastní klíč, nelze spustit aplikaci jen jednou a mít kód pro všechny uživatele.

6.4.2 Dostupné autentizační moduly

Po konfiguraci jednotlivých variant MFA máme k dispozici několik autentizačních modulů, které můžeme využívat nejen v konfiguračním souboru /etc/pam.d/common-auth, který slouží pro nastavení společné autentizace pro všechny služby, ale i pro autentizaci pouze konkrétních aplikací a služeb. Moduly můžeme také využít pro vytváření nových variant MFA.

```
pam_unix.so
pam_google_authenticator.so
pam_fprintd.so
pam_u2f.so
```

6.4.3 Možnosti vícefaktorové autentizace

V adresáři /etc/pam.d nalezneme konfigurační soubory jednotlivých aplikací a služeb, kde můžeme vložit a kombinovat jednotlivé autentizační moduly a vytvořit tak vícefaktorovou autentizaci pro konkrétní aplikaci či službu. Můžeme například nastavit vícefaktorovou autentizaci pro Xscreensaver, což je spořič obrazovky, který dokáže uzamknout zařízení a následně vyžaduje autentizaci. Na obrázku 6.15 můžeme vidět konfigurační soubor /etc/pam.d/xscreensaver pro Xscreensaver. Soubor původně obsahoval pouze řádky:

```
@include common-auth
@include common-account
```

Tyto řádky zakomentujeme nebo smažeme, protože pro Xscreensaver nechceme využívat autentizaci, která je nastavena v souboru /etc/pam.d/common-auth, ale chceme mu vytvořit vlastní autentizační schéma. Z dostupných autentizačních modulů si můžeme vytvořit například novou variantu MFA, kterou jsme si ještě nevytvářeli. Jako například vícefaktorovou autentizaci v kombinaci heslo, Google Authenticator a YubiKey.

```
auth    requisite  pam_unix.so
auth    requisite  pam_google_authenticator.so
auth    requisite  pam_u2f.so cue
```

Listing 6.21: Heslo, Google Authenticator a YubiKey

Heslo, Google Authenticator a otisk prstu.

```
auth    requisite  pam_unix.so
auth    requisite  pam_google_authenticator.so
auth    requisite  pam_fprintd.so
```

Listing 6.22: Heslo, Google Authenticator a otisk prstu

YubiKey a otisk prstu.

```
auth    requisite  pam_u2f.so cue
auth    requisite  pam_fprintd.so
```

Listing 6.23: YubiKey a otisk prstu

Nebo v konfiguračním souboru můžeme nastavit autentizaci pomoci všech čtyř autentizačních modulu viz. obrázek 6.15. Tato MFA určitě nebude uživatelsky nejpřívětivější a může nastat i problém s množstvím periferních zařízení, které pro autentizaci potřebujeme připojit.

```
 GNU nano 3.2                    /etc/pam.d/xscreensaver

# /etc/pam.d/xscreensaver - PAM behavior for xscreensaver

#@include common-auth
#@include common-account

auth requisite pam_unix.so
auth requisite pam_fprintd.so
auth requisite pam_google_authenticator.so
auth requisite pam_u2f.so cue
```

Obrázek 6.15: Konfigurační soubor pro Xscreensaver

6.5 Zhodnocení jednotlivých variant autentizace

Každá z jednotlivých variant vícefaktorové autentizace má určitě své plusy a mínusy, ať už jde o rychlost nebo bezpečnost autentizace a s tím související počet použitých autentizačních faktorů nebo cena případných doplňků a zařízení, které se rozhodneme pro autentizaci využít. Je tedy vždy dobré zvážit, jakou variantu vícefaktorové autentizace se rozhodneme implementovat.

Varianta a - Tato varianta MFA využívá tří autentizačních faktoru a díky tomu by se dala označit za nejvíce bezpečnou ze všech variant. Jako překážka se může jevit pořizovací cena dvou zařízení potřebných pro autentizaci (čtečka otisku prstu Upek eikon a YubiKey 5 NFC), kde navíc každé z těchto zařízení vyžaduje USB konektor a ty jsou na zařízení Raspberry Pi 4 Model B pouze čtyři. Implementace je poměrně snadná, jen je potřeba být ostražitý při vytváření hesla s pomoci YubiKey a zvoleném typu klávesnice. Dále je potřeba zmínit chybovost čtečky otisku prstu Upek eikon, která může celkovou autentizaci značně zpomalovat.

Varianta b - Tato varianta MFA využívá pouze dvou autentizačních faktorů a dala by se tedy označit za méně bezpečnou než variantu a. V této variantě MFA je opět využit hardwarový token YubiKey 5 NFC, ale trochu jiným způsobem, protože využívá svůj vlastní autentizační modul. Implementace této varianty MFA je poměrně snadná a funguje bez problému. MFA je uživatelsky přívětivá a poměrně rychlá.

Varianta c - U této varianty MFA využíváme dvou autentizačních faktorů a poprvé zde není využito faktoru znalosti. Tato varianta MFA by mohla být z hlediska bezpečností srovnatelná s variantou b. Opět však nastává problém s připojením dvou zařízení pomoci USB. Tato MFA může na uživatele působit poměrně atraktivně, vzhledem k autentizačním metodám, které jsou použity. Implementace této varianty MFA je také poměrně snadná, jen je opět potřeba dát pozor na zvolený typ klávesnice v zařízení Raspberry Pi, při používání RFID čtečky. Pokud nenastane chyba při ověření otisku prstu, je tato MFA velmi rychlá.

Kapitola 7

MFA pro přístup ke službám

7.1 Návrh MFA pro SSH/SFTP - varianta a

Návrh varianty MFA pro zabezpečení vzdáleného přístupu k zařízení Raspberry Pi, ve složení:

```
Heslo (znalost)
Softwarový token Google Authenticator (vlastnictví)
```

PAM modul a aplikaci v mobilním telefonu jsme si již nastavili v kapitole 6.4, nyní přejdeme k samotné implementaci vícefaktorové autentizace pro službu SSH pomoci modulu PAM. Díky tomu, že neprovádím konfiguraci SSH přes SSH, ale mám k Raspberry Pi připojen vlastní monitor, myš a klávesnici, nemusím mít obavy, že si vinou chybné konfigurace SSH nezvratně odříznu cestu k zařízení Raspberry Pi. V opačném případě je dobré neuzavírat počáteční připojení přes SSH, kdybychom náhodou udělali v konfiguraci chybu a nemohli se již k zařízení připojit.

Otevřeme si konfigurační soubor PAM pro SSH v textovém editoru nano pomoci příkazu v terminálu.

```
pi@raspberrypi:~ $ sudo nano /etc/pam.d/sshd
```

Standardní autentizace pomoci hesla je již v tomto konfiguračním souboru nastavena díky řádku '@include common-auth', pokud jsme tedy v souboru /etc/pam.d/common-auth neprováděli žádné změny týkající se autentizace, jako například v předchozí kapitole.

Nyní tedy stačí pouze přidat modul pro autentizaci pomoci Google Authenticator, tedy modul pam_google_authenticator.so, který přidáme před nebo až za řádek '@include common-auth' v souboru /etc/pam.d/sshd, podle toho v jakém pořadí chceme autentizační moduly volat.

Na obrázku 7.1 můžeme vidět, že jsem tento nový modul pro autentizaci pomoci Google Authenticator přidal až za řádek '@include common-auth', syntaxí daného řádku s tímto autentizačním modulem říkáme, že vyžadujeme ověření pomoci modulu pam_google_authenticator.so a tedy bez úspěchu tohoto modulu nemůže být celková autentizace úspěšná.

```
 GNU nano 3.2                                /etc/pam.d/sshd

# PAM configuration for the Secure Shell service
# Standard Un*x authentication.
@include common-auth
auth required pam_google_authenticator.so
```

Obrázek 7.1: Konfigurační soubor /etc/pam.d/sshd

Jako další krok je potřeba provést drobnou úpravu v následujícím konfiguračním souboru.

/etc/ssh/sshd_config

Je potřeba upravit následující řádek v tomto souboru ze tvaru:

```
ChallengeResponseAuthentication no
```

na tvar:

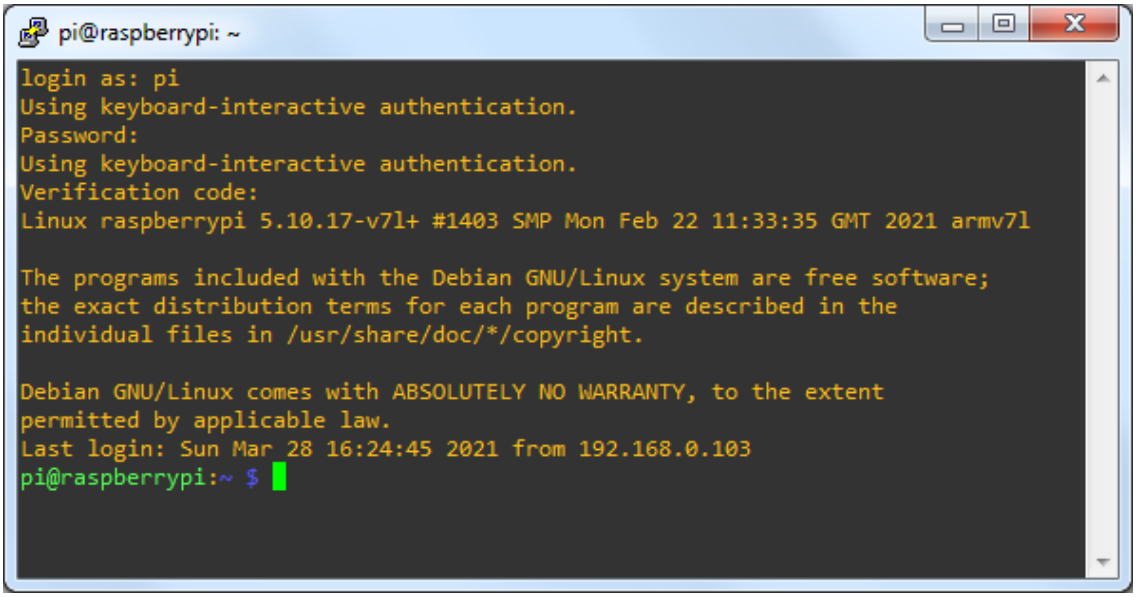
```
ChallengeResponseAuthentication yes
```

Nyní restartujeme službu SSH, aby se projevilo nové nastavení.

```
pi@raspberrypi:~ $ sudo service ssh restart
```

Listing 7.1: Restart služby SSH

Po konfiguraci jednotlivých souborů a restartování služby SSH můžeme ověřit MFA pomoci jiného počítače a SSH klienta PuTTY. Na obrázku 7.2 můžeme vidět, že u autentizace bylo potřeba zadat kromě hesla také ověřovací kód z aplikace Google Authenticator, kterou máme v mobilním telefonu.



Obrázek 7.2: Ověření MFA v PuTTY - varianta a

Na obrázku 7.3 můžeme vidět výpis souboru /var/log/auth.log ze zařízení Raspberry Pi, kde můžeme pozorovat průběh autentizace se zapojením modulu pam_google_authenticator.so.

```
Apr 19 22:12:31 raspberrypi sshd(pam_google_authenticator)[1790]: debug: start of google_authenticator for "pi"
Apr 19 22:12:31 raspberrypi sshd(pam_google_authenticator)[1790]: debug: Secret file permissions are 0400. Allowed permissions are 0600
Apr 19 22:12:31 raspberrypi sshd(pam_google_authenticator)[1790]: debug: "/home/pi/.google_authenticator" read
Apr 19 22:12:31 raspberrypi sshd(pam_google_authenticator)[1790]: debug: shared secret in "/home/pi/.google_authenticator" processed
Apr 19 22:12:43 raspberrypi sshd(pam_google_authenticator)[1790]: debug: no scratch code used from "/home/pi/.google_authenticator"
Apr 19 22:12:43 raspberrypi sshd(pam_google_authenticator)[1790]: Accepted google_authenticator for pi
Apr 19 22:12:43 raspberrypi sshd(pam_google_authenticator)[1790]: debug: "/home/pi/.google_authenticator" written
Apr 19 22:12:43 raspberrypi sshd[1788]: Accepted keyboard-interactive/pam for pi from 192.168.0.103 port 50437 ssh2
Apr 19 22:12:43 raspberrypi sshd[1788]: pam_unix(sshd:session): session opened for user pi by (uid=0)
Apr 19 22:12:43 raspberrypi systemd-logind[421]: New session 11 of user pi.
```

Obrázek 7.3: Výpis souboru /var/log/auth.log

7.2 Návrh MFA pro SSH/SFTP - varianta b

Návrh další varianty MFA pro SSH. Tuto variantu MFA bude tvořit:

`Heslo (znalost)`
`Hardwarový token YubiKey 5 NFC (vlastnictví)`

Abychom mohli využít zařízení YubiKey 5 NFC při autentizace u SSH, tedy v případě, kdy není toto zařízení fyzicky připojeno k zařízení Raspberry Pi, ale k jinému zařízení, musíme použít jiný modul než pam_u2f.so, ten totiž pro SSH použít nelze. Nainstalujeme tedy nový PAM modul pam_yubico.so pomoci následujícího příkazu:

```
pi@raspberrypi:~ $ sudo apt install libpam-yubico
```

Listing 7.2: Instalace libpam-yubico

Dále si vytvoříme speciální soubor.

```
pi@raspberrypi:~ $ sudo nano /etc/ssh/authorized_yubikeys
```

Tento soubor bude obsahovat uživatelské jméno a ID tokenu YubiKey oddělené dvojtečkami (stejný formát jako soubor passwd) pro každého uživatele, kterého chcete povolit. ID tokenu lze získat z našeho YubiKey 5 NFC například při vygenerování jednorázového hesla OTP, toto jednorázové heslo se vygeneruje při krátkém přiložení prstu na kovový kontakt YubiKey 5 NFC. ID tokenu je tvořeno pouze prvními 12 znaky tohoto OTP.

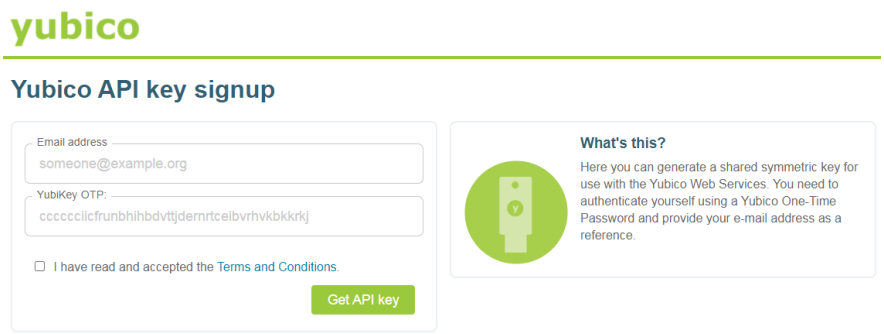
```
pi:ccccccctjuvng
```

Listing 7.3: Soubor /etc/ssh/authorized_yubikeys.

Dalším krokem je vygenerování sdíleného symetrického klíče, pro použití YubiKey s webovými službami Yubico, na následující adrese, kde po zadání e-mailové adresy a vygenerování OTP získáme *Client ID* a *Secret key*, které využijeme později v konfiguračním souboru /etc/pam.d/sshd.

```
https://upgrade.yubico.com/getapikey
```

Listing 7.4: Adresa pro registraci Yubico API key.



Obrázek 7.4: Webová stránka https://upgrade.yubico.com/getapikey

Otevřeme si konfigurační soubor /etc/pam.d/sshd a zde implementujeme autentizaci pomoci modulu pam_yubico.so. Jako argumenty tohoto modulu přidáme *id* a *key*, tyto argumenty reprezentují *Client ID* a *Secret key*, které jsme získali v předchozím kroku registrace na webové stránce. Kromě těchto dvou argumentů je také potřeba přidat argument *authfile*, který označuje umístění souboru, obsahujícího mapování ID tokenů YubiKey na uživatelská jména. Na obrázku 7.5 lze vidět příklad této implementace.

```
 GNU nano 3.2                                /etc/pam.d/sshd

# PAM configuration for the Secure Shell service
# Standard Un*x authentication.
@include common-auth
auth required pam_yubico.so id=63050 key=Ffd+CifKA0EYfLfFoPCFpe0yZ0s= authfile=/etc/ssh/authorized_yubikeys
```

Obrázek 7.5: Ukázka souboru /etc/pam.d/sshd

Dále je potřeba se opět ujistit, že v souboru /etc/ssh/sshd_config je následující řádek ve tvaru *yes* a ne náhodou *no*, jak už bylo zmíněno a popsáno v předchozí variantě a.

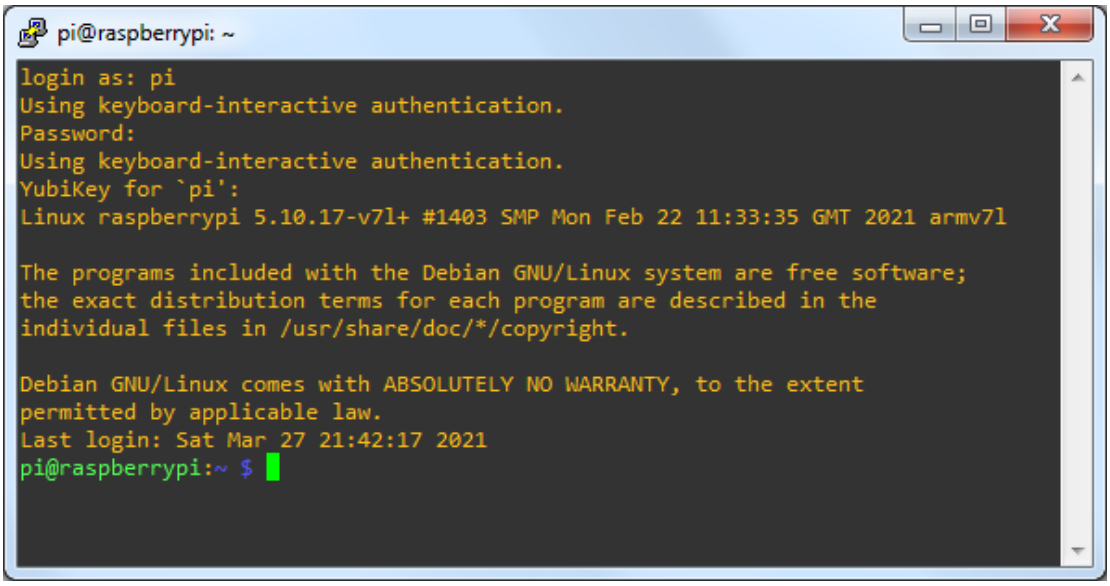
```
ChallengeResponseAuthentication yes
```

Pro to, aby se projevila nová implementace MFA, restartujeme službu SSH.

```
pi@raspberrypi:~ $ sudo service ssh restart
```

Listing 7.5: Restart SSH

Vícefaktorovou autentizaci opět ověříme pomoci jiného počítače a SSH klienta PuTTY. Na obrázku 7.6 můžeme vidět, že při autentizaci byl kromě hesla vyžadován také hardwarový token YubiKey. Pro úspěšné ověření pomoci YubiKey, bylo nutné se dotknout kovového kontaktu na zařízení YubiKey 5 NFC a tím vygenerovat kód, nestačilo mít zařízení pouze připojené k počítači.



Obrázek 7.6: Ověření MFA v PuTTY - varianta b

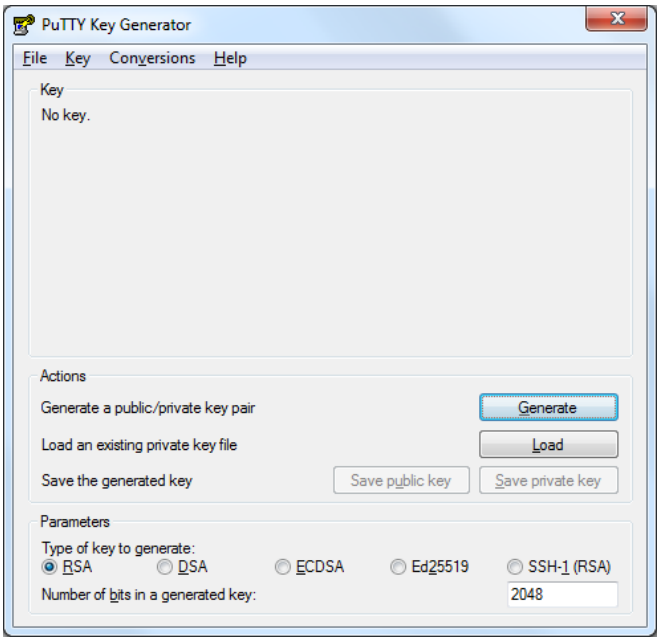
7.3 Návrh MFA pro SSH/SFTP - varianta c

Implementace MFA pro uživatele, kteří chtějí přistoupit přes protokol SFTP nebo SSH na server (Raspberry Pi). Zaroveň si v této variantě ukážeme, jak uzamknout uživatele v určitém adresáři a nepustit ho do systémových souborů, kam by neměl mít přístup. Tato varianta MFA bude obsahovat:

```
Heslo (znalost)
SSH klíče (vlastnictví)
```

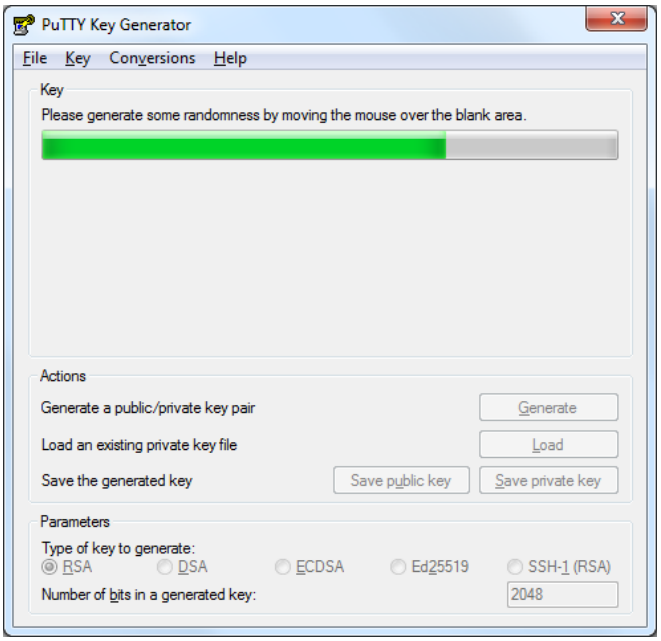
SSH klíče fungují tak, že si vygenerujeme dvojici klíčů, jeden soukromý a druhý veřejný. Soukromý klíč si ponecháme na klientském zařízení a veřejný klíč přesuneme do Raspberry Pi. Tyto klíče slouží k identifikaci sebe sama na serveru pomoci kryptografie veřejného klíče a autentizace výzvou-odpovědí. Na server se bude moci přihlásit pouze ten, kdo má tento soukromý klič.

Protokol SFTP (SSH File Transfer Protocol) je založen na SSH a pro jeho použití stačí mít v počítači nainstalován balíček OpenSSH. Pro přenos souboru se využívá stejného tunelu, jako pro přistup ke vzdálené konzoli. SFTP protokol poskytuje široké spektrum možností pro operace se soubory, lze ho brát jako zjednodušený, vzdálený souborový systém. Umožňuje spravovat data na serveru, jako je vypisování adresářů nebo vytváření, upravování a odstraňování souborů. Pro připojení na server (Raspberry Pi) pomoci protokolu SFTP využijeme populárního klienta WinSCP, který je nainstalován na klientském počítači. WinSCP je volně dostupný SFTP klient pro operační systém Windows a jeho funkcí je bezpečný přenos souborů mezi lokálním a vzdáleným zařízením. Na klientském zařízení s WinSCP spustíme *PuTTY Key Generator*, to provedeme tak, že po spuštění WinSCP klikneme na možnost *Nástroje* a z nabídky vybereme *Spustit PuTTYgen*.



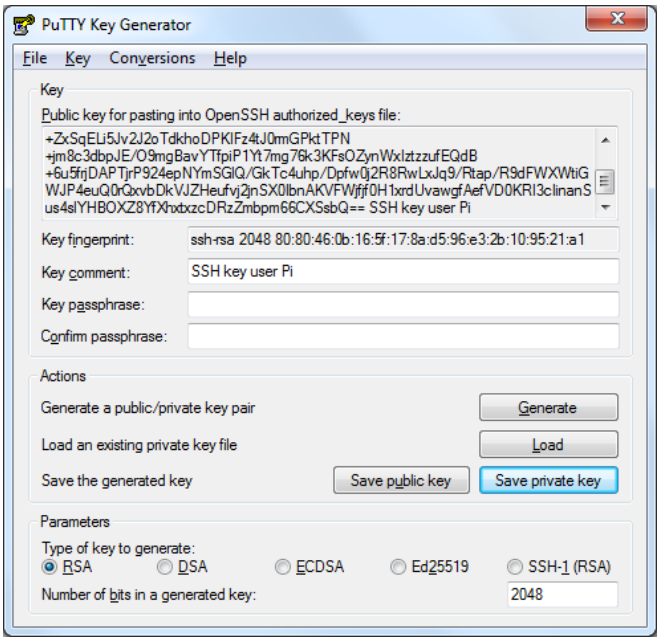
Obrázek 7.7: PuTTY Key Generator

Na obrázku 7.7 můžeme vidět úvodní okno tohoto programu, kde nastavíme typ klíče na *RSA*, počet bitů 2048 a klikneme na možnost *Generate*. Na Obrázku 7.8 lze vidět, že při generování klíčů jsme požádáni, abychom pohybovali myší v určeném prostoru a tím zajistili, že vygenerované klíče budou unikátní.



Obrázek 7.8: Vytváření klíčů v PuTTY Key Generator

Nyní, když máme vygenerované SSH klíče, nastavíme těmto klíčům název v poli *Key comment*, viz. obrázek 7.9. Tímto názvem se bude později klíč hlásit při autentizaci. Jako další můžeme nastavit přístupový kód k soukromému klíči v poli *Key passphrase*. Tento kód bude vyžadován při použití soukromého klíče. Pro lepší zabezpečení soukromého klíče se doporučuje tento kód nastavit, my ho však nastavovat nebudeme a ponecháme pole prázdné. Nyní si pomoci možnosti *Save private key* uložíme náš soukromý klíč na bezpečné místo v počítači, klíč musí mít příponu .ppt. Jako poslední zkopírujeme veřejný klíč, který je uveden nahoře v textovém poli a později tento veřejný klíč vložíme do vytvořeného souboru s veřejnými klíčí na zařízení Raspberry Pi.



Obrázek 7.9: Vygenerované klíče v PuTTY Key Generator

Na serveru Raspberry Pi je potřeba provést některé příkazy k nastavení souboru /home/-pi/.ssh/autherized_keys. Tento soubor bude ověřovat SSH deamon, když bude k autentizaci použit soukromý klíč. Pomoci následujících příkazů vytvoříme složku s potřebným oprávněním a vytvoříme soubor /home/pi/.ssh/authorized_keys. Do tohoto souboru pak vložíme náš veřejný klíč. SSH ověří všechny soukromé klíče proti přítomnému veřejnému klíči.

```
pi@raspberrypi:~ $ sudo install -d -m 700 ~/.ssh
pi@raspberrypi:~ $ sudo nano ~/.ssh/authorized_keys
```

```
 GNU nano 3.2                          /home/pi/.ssh/authorized_keys

ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAsfGXyHTg1ZbEXWDcd0QfLFQlV6cniAKeZQ8NdBgdna+gB39WyIhnB13Pwvda/h5prTviJetE+Zx$
```

Obrázek 7.10: Obsah souboru /home/pi/.ssh/authorized_keys

Jako další věc je potřeba pomoci následujících příkazů nastavit správné oprávnění souboru, aby ho mohl SSH při autentizaci číst.

```
pi@raspberrypi:~ $ sudo chmod 644 ~/.ssh/authorized_keys
pi@raspberrypi:~ $ sudo chown pi:pi ~/.ssh/authorized_keys
```

Nyní v textovém editoru nano otevřeme konfigurační soubor /etc/ssh/sshd_config.

```
pi@raspberrypi:~ $ sudo nano /etc/ssh/sshd_config
```

V tomto konfiguračním souboru je potřeba přidat následující řádek, kterým říkáme, že pro autentizaci je vyžadován klíč a zároveň heslo.

```
AuthenticationMethods publickey, password
```

Tímto máme MFA pro uživatele pi využívajícího protokol SSH/SFTP nastavenu, uživatel pi však není na SFTP serveru nijak omezen a může procházet všechny adresáře. Budeme tedy chtít uživatele uzavřít v adresáři /home, kde bude mít přístup pouze do svého adresáře a adresářů ostatních uživatelů. Toho lze dosáhnout poměrně snadno, díky tomu, že OpenSSH umožňuje automatickou tvorbu chrootu dle potřeby a obsahuje integrovaný SFTP server. Toto nastavení se provádí v konfiguračním souboru /etc/ssh/sshd_config, nejprve však vytvoříme novou skupinu sftpgroup pomoci následujícího příkazu:

```
pi@raspberrypi:~ $ sudo groupadd sftpgroup
```

Do této skupiny přiřadíme uživatele pi.

```
pi@raspberrypi:~ $ sudo usermod -G sftpgroup pi
```

Nyní na konec konfiguračního souboru /etc/ssh/sshd_config vložíme následnou konfiguraci pro omezení práv uživatele a uložíme.

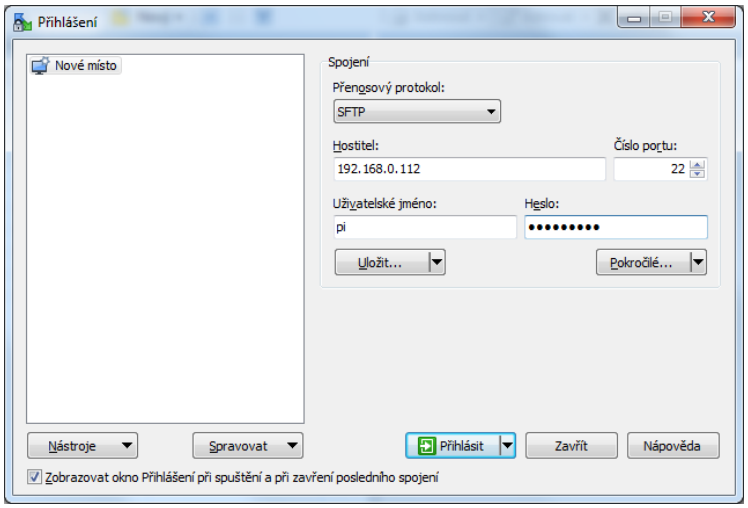
```
Subsystem sftp internal-sftp

  Match Group sftpgroup
  ChrootDirectory /home
  ForceCommand internal-sftp
  X11Forwarding no
  AllowTcpForwarding no
```

Restartujeme službu SSH, aby se projevily nově nastavené změny.

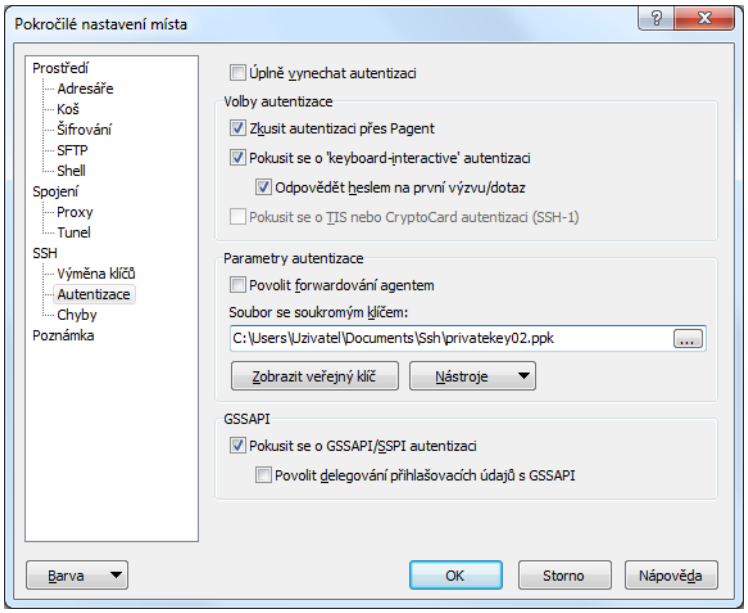
```
pi@raspberrypi:~ $ sudo service ssh restart
```

Nyní by měly být patrné námi provedené změny, pokusíme se tedy připojit přes SFTP klienta WinSCP na SFTP server (Raspberry Pi). Na obrázku 7.11 lze vidět přihlašovací okno ve WinSCP,

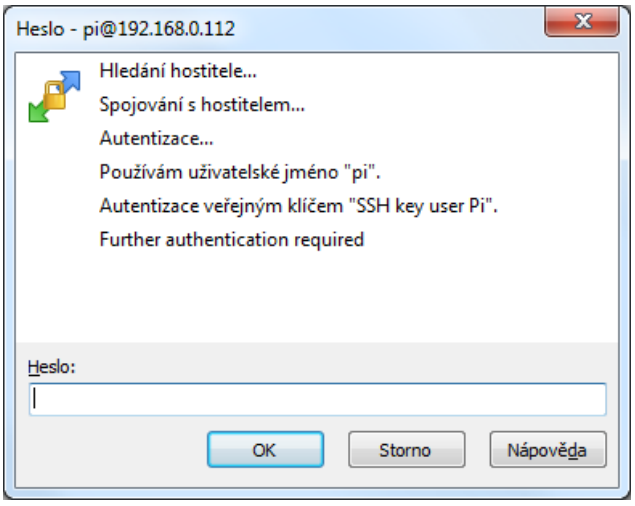
v tomto okně zvolíme protokol SFTP, zadáme adresu Raspberry Pi, která je 192.168.0.112, uživatelské jméno a heslo. Pro to, aby byla autentizaci uživatele úspěšná, je potřeba vložit soukromý klíč, to provedeme tak, že klikneme na možnost *Pokročilé* a z nabídky vlevo klikneme na *Autentizace*, poté zadáme cestu k našemu soukromému klíči viz. obrázek 7.12. Kdybychom se pokusili přihlásit bez vložení soukromého klíče autentizace okamžitě selže. V případě hesla, můžeme toto heslo zadat před pokusem o přihlášení nebo až v průběhu autentizace.



Obrázek 7.11: Přihlášení ve WinSCP



Obrázek 7.12: Vložení soukromého klíče ve WinSCP



Obrázek 7.13: Průběh autentizace ve WinSCP

Na obrázku 7.13 můžeme vidět průběh autentizace, kde si lze všimnout, že uživatel pi byl ověřen veřejným klíčem s názvem *SSH key user Pi*, který jsme si zadali na obrázku 7.9 a nyní se čeká na zadání hesla. Na obrázku 7.14 lze vidět výpis z /var/log/auth.log při pokusu o přihlášení bez vložení soukromého klíče do WinSCP.

```
raspberrypi sshd[1641]: Connection closed by authenticating user pi 192.168.0.103 port 62169 [preauth]
```

Obrázek 7.14: Výpis souboru /var/log/auth.log, bez vložení soukromého klíče

Ve výpisu na obrázku 7.15 byl při přihlášení soukromý klíč vložen, ale při pokusu o přihlášení bylo zadáno nesprávné heslo a autentizace nebyla úspěšná, což můžeme vidět v prvních třech řádcích výpisu. Při opakovaném zadaní hesla již bylo heslo zadáno správné a autentizace byla úspěšná viz. poslední tři řádky výpisu.

```
raspberrypi sshd[1667]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser=
rhost=192.168.0.103  user=pi
raspberrypi sshd[1667]: Failed password for pi from 192.168.0.103 port 62185 ssh2
raspberrypi sshd[1667]: Accepted password for pi from 192.168.0.103 port 62185 ssh2
raspberrypi sshd[1667]: pam_unix(sshd:session): session opened for user pi by (uid=0)
raspberrypi systemd-logind[362]: New session 22 of user pi.
```

Obrázek 7.15: Výpis souboru /var/log/auth.log, s vloženým soukromým klíčem

7.4 Zhodnocení jednotlivých variant autentizace

Varianta a - Výhoda této MFA spočívá v tom, že jako druhý autentizační faktor využíváme mobilní telefon, který nosí většina lidí neustále u sebe a uživatel tak není povinen u sebe nosit další speciální autentizační předmět. Tímto odpadají také náklady spojené s pořízením specializovaného

zařízení. V případě ztráty nebo poruchy mobilního telefonu, může uživatel pro přístup využít kódy pro obnovení. Implementace této varianty MFA je poměrně snadná a spolehlivě funguje.

Varianta b - MFA v kombinaci s hardwarovým klíčem YubiKey 5 NFC je opak varianty a, uživatel u sebe musí nosit další speciální předmět pouze pro autentizaci a pořizovací náklady YubiKey 5 NFC nejsou zrovna zanedbatelné. V případě ztráty tokenu se již uživatel nebude moci přihlásit. Implementace této varianty je mírně obtížnější oproti variantě a.

Varianta c - U této varianty MFA s SSH klíči může nastat problém s bezpečným přenosem soukromého klíče, pokud by se chtěl uživatel připojit z jiného zařízení než, na kterém má svůj soukromý klíč uložen. Tato varianta nevyžaduje žádné pořizovací náklady a v případě dodržení bezpečnostních opatření, nehrozí ztráta soukromého klíče. Složitost implementace je srovnatelná s variantou b.

Kapitola 8

Závěr

Cílem diplomové práce bylo přehledně zpracovat, navrhnout a ověřit vícefaktorovou autentizaci v embedded zařízení typu Raspberry Pi. V teoretické části jsem nejprve popsal metody autentizace, kde jsem se konkrétněji zaměřil na metody a zařízení, které jsou následně použity v praktické části této diplomové práce. V další části teorie jsem vysvětlil pojem vícefaktorová autentizace a pojmy s ní spojené. Následoval popis struktury a vysvětlení fungování PAM, kde je také objasněna syntaxe konfiguračních souborů a možnost autentizace pomoci jednotlivých autentizačních modulů. V poslední části teorie je vysvětlen pojem embedded zařízení a představeno několik zástupců, na kterých by případně mohla být realizována implementace vícefaktorové autentizace. Vzhledem k tomu, že škola pořídila několik nových zařízení Raspberry Pi 4 Model B, využil jsem této možnosti a zapůjčil si jedno z těchto zařízení, na kterém jsem také realizoval praktickou část této diplomové práce.

Hlavní náplň práce tedy spočívala v praktické části, kde jsem navrhnul a ověřil několik variant vícefaktorové autentizace pro embedded zařízení Raspberry Pi. Tyto návrhy jsou rozděleny do dvou kapitol, podle způsobu přístupu k zařízení Raspberry Pi. V první z těchto kapitol jsou navrženy a ověřeny varianty MFA, které jsou vhodné pro zařízení Raspberry Pi v režimu klasické stolního počítače a ve druhé z těchto kapitol jsou navrženy varianty MFA pro vzdálený přístup k zařízení pomoci SSH nebo SFTP. Pro každou z těchto možností přístupu, bylo vhodné použit jiné autentizační metody, například z důvodu potřebných periferních zařízení a jejich možnosti nasazení na klientském počítači, v případě přístupu přes SSH/SFTP. K uživateli se lze také chovat trochu jinak, když přistupuje k zařízení přes SSH a úspěšně projde procesem MFA, než v případě fyzického přístupu k tomuto zařízení, který může využívat třeba více uživatelů.

Při návrhu a konfiguraci jednotlivých autentizačních metod a veškeré činnosti s tím související, jsem byl k zařízení Raspberry Pi vždy připojen jako ke klasickému počítači, kde jsem měl připojeny všechny periferní zařízení, jako je myš, klávesnice a monitor. Vzdálený přístup k zařízení přes SSH jsem použil pouze v případě testování MFA u protokolu SSH/SFTP, ne však pro konfiguraci. Tato možnost mi poskytovala více volnosti při konfiguraci a testovaní MFA u SSH a nemusel jsem se obávat případné chyby při konfiguraci, která by mi mohla znemožnit přístup přes SSH, na kterém

bych byl závislý. Při konfiguraci MFA a další potřebné činnosti s tím související, jsem se u embedded zařízení Raspberry Pi 4 Model B a operačního systému Raspberry Pi OS (32-bit) nesetkal s žádným omezením v rámci operačního systému, PAM nebo dostupností aplikací či služeb. Částečně omezující mohla být pouze velikost RAM (1 GB), ale nijak závažně jsem to nepociťoval. Z hlediska možných doplňků a dostupných rozhranní, jako je SPI nebo I2C, poskytuje toto zařízení naopak zajímavé možnosti z pohledu autentizace.

V této práci jsou popsány návrhy a metody autentizace, které se povedlo úspěšně realizovat, nemalé množství času a úsilí však také zabraly pokusy, které se realizovat nepovedlo a v textu této diplomové práce se o nich nezmiňuji, ale za zmínku určitě stojí. Jedná se například o možnost autentizace pomoci RFID čtečky RC522, tuto čtečku můžeme připojit k Raspberry Pi pomoci sériového rozhranní SPI. Pomoci skriptu v Pythonu, lze s touto čtečkou jednoduše komunikovat a provádět operace čtení a zápisu, z nebo na RFID tag. Po počátečních úspěších jsem chtěl tuto čtečku zapojit do procesu autentizace, ale po několika pokusech, otestování nejrůznějších RFID projektu na autentizaci a dokonce propojení této RFID čtečky RC522 s mikrokontrolérem Arduino Uno, se autentizaci zprovoznit nepodařilo. Zvolil jsem proto jinou, mnohem omezenější variantu RFID čtečky, kterou využívám v této diplomové práci. Tato čtečka zdaleka nedosahuje možností RFID čtečky RC522, ale konkrétně pro jednoduchou autentizaci pomoci RFID karty zcela postačuje. Také bych chtěl vyzdvihnout univerzálnost použití modulu pam_unix.so, který slouží primárně k ověření uživatele pomoci uživatelského hesla (znalost), ale lze využít i pro autentizaci pomoci RFID karty nebo YubiKey (statické heslo), jak lze vidět v návrzích MFA. Z použitých zařízení pro autentizaci bych nejvíce vyzdvihl hardwarový token YubiKey 5 NFC, který vždy spolehlivě fungoval a vzhledem k jeho širokým možnostem využití, toto zařízení určitě stojí za pozornost. Pokud však uživatel nechce utrácet peníze za hardwarový token nebo jiné vybavení, velmi dobře se osvědčil také softwarový token Google Authenticator, v podobě aplikace na mobilním telefonu.

Po zkušenostech, které jsem nasbíral při konfiguraci a testování MFA, musím přiznat, že s konfiguračními soubory PAM, není vždy radno si zahrávat. Případná větší chyba v konfiguraci, vás totiž může zcela odříznout od zařízení. Tato situace se mi přihodila jednou, když jsem udělal chybu v konfiguračním souboru /etc/pam.d/common-auth a v kombinaci s nešťastným nastavením spořiče obrazovky Xscreensaver, byl problém na světě. Tuto situaci jsem musel řešit nahráním nového operačního systému na microSD kartu.

Věřím, že tato diplomová práce přináší přehledné zpravování některých možností vícefaktorové autentizace, které je možné implementovat na embedded zařízení Raspberry Pi 4 Model B a pomůže v případné implementaci MFA na další zařízení.

Literatura

1. KAMENÍČKOVÁ, Petra. *Implementace externích autentizačních modulů pro nginx*. Brno, 2015. Bakalářská práce. Vysoké učení technické v Brně.
2. DRAŠAR, Martin. *Password Based Authentication [online]*. 2009 [cit. 2021-04-28]. Dostupné také z: `https://is.muni.cz/th/qkn59/thesis.pdf`. Diplomová práce. Masarykova univerzita, Fakulta informatiky, Brno. Vedoucí práce Zdeněk ŘÍHA.
3. TICHÝ, Jan. *Ukládání hesel v databázi* [online] [cit. 2021-03-07]. Dostupné z: `https://www.phpguru.cz/clanky/hashovani-hesel`.
4. KODYS.CZ. *Radiofrekvenční identifikace - RFID* [online] [cit. 2021-03-07]. Dostupné z: `https://www.kodys.cz/technologie/rfid`.
5. *RFID v kostce* [online] [cit. 2021-04-17]. Dostupné z: `https://www.soselectronic.cz/articles/sos-supplier-of-solution/rfid-v-kostce-2349`.
6. HUMPOLÍK, Jan. *Webová aplikace využívající vícefaktorovou autentizaci*. Brno, 2013. Diplomová práce. Vysoké učení technické v Brně.
7. YUBICO.COM. *How the YubiKey works* [online] [cit. 2021-03-07]. Dostupné z: `https://www.yubico.com/why-yubico/how-the-yubikey-works/`.
8. YUBIKEY.CZ. *Co je to YubiKey* [online] [cit. 2021-03-17]. Dostupné z: `https://www.yubikey.cz/`.
9. *OTPs Explained* [online] [cit. 2021-03-17]. Dostupné z: `https://developers.yubico.com/OTP/OTPs_Explained.html`.
10. YUBICO.COM. *What is FIDO U2F?* [Online] [cit. 2021-03-17]. Dostupné z: `https://www.yubico.com/authentication-standards/fido-u2f/`.
11. *What is OATH?* [Online] [cit. 2021-03-17]. Dostupné z: `https://developers.yubico.com/OATH/`.
12. *What is PGP?* [Online] [cit. 2021-03-17]. Dostupné z: `https://developers.yubico.com/PGP/`.

13. NICOLLS, Dean. *What is Biometric Authentication?* [Online] [cit. 2021-03-07]. Dostupné z: https://www.jumio.com/what-is-biometric-authentication/.

14. AUTOIDINDIA.COM. *Different Types of Fingerprint Scanners – Optical, Capacitive and Ultrasonic* [online] [cit. 2021-03-07]. Dostupné z: https://autoidindia.com/different-types-of-fingerprint-scanners-optical-capacitive-and-ultrasonic/.

15. *Types of Biometrics* [online] [cit. 2021-03-07]. Dostupné z: https://www.biometricsinstitute.org/what-is-biometrics/types-of-biometrics/.

16. ONELOGIN. *What is Multi-Factor Authentication (MFA)?* [Online] [cit. 2021-04-17]. Dostupné z: https://www.onelogin.com/learn/what-is-mfa.

17. TATAM, Robin. *What's the Difference Between Two-Factor Authentication and Multi-Factor Authentication?* [Online] [cit. 2021-04-17]. Dostupné z: https://www.helpsystems.com/resources/articles/whats-difference-between-two-factor-authentication-and-multi-factor.

18. ANDREW G. MORGAN, Thorsten Kukuk. *The Linux-PAM System Administrators' Guide* [online] [cit. 2021-03-10]. Dostupné z: http://linux-pam.org/Linux-PAM-html/Linux-PAM_SAG.html.

19. *Pluggable Authentication Modules (PAM)* [online] [cit. 2021-03-10]. Dostupné z: http://web.mit.edu/rhel-doc/5/RHEL-5-manual/Deployment_Guide-en-US/ch-pam.html.

20. BOBČÍK, BOLESLAV. *PAM - správa autentizačních mechanismů* [online] [cit. 2021-03-10]. Dostupné z: https://www.root.cz/clanky/pam-sprava-autentizacnich-mechanismu/.

21. WIKIPEDIE, Přispěvatelé. *Vestavěný systém* [online] [cit. 2021-03-14]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Vestav%C4%9Bn%C3%BD_syst%C3%A9m&oldid=18015612.

22. TECHTARGET. *embedded device* [online] [cit. 2021-03-14]. Dostupné z: https://whatis.techtarget.com/definition/embedded-device.

23. HUGHES, Owen. *10 Raspberry Pi alternatives for you to try out* [online] [cit. 2021-04-17]. Dostupné z: https://www.techrepublic.com/article/10-raspberry-pi-alternatives-for-you-to-try-out/#listicle-listicle-049d0624-1b5c-4f1c-9d72-33ac328d4ea3.

24. VÍTEK, Jan. *Rock Pi 4 Model C nabízí i podporu NVMe a eMMC* [online] [cit. 2021-04-17]. Dostupné z: https://www.svethardware.cz/rock-pi-4-model-c-nabizi-i-podporu-nvme-a-emmc/52697.

25. ALL3DP. *2021 Best Single Board Computers / Raspberry Pi Alternatives* [online] [cit. 2021-04-17]. Dostupné z: https://all3dp.com/1/single-board-computer-raspberry-pi-alternative/.

26. *BPI-M5 Specification* [online] [cit. 2021-04-17]. Dostupné z: http://www.banana-pi.org/m5.html.

27. RPISHOP.CZ. *Raspberry Pi 4 Model B - 1GB RAM* [online] [cit. 2021-03-14]. Dostupné z: https://rpishop.cz/prodej-ukoncen/1600-RPI401-7657569311684.html.

28. TECHPOWERUP.COM. *Upek Eikon* [online] [cit. 2021-03-17]. Dostupné z: https://www.techpowerup.com/review/upek-eikon/3.html.

29. *Čtečka kódů RFID, frekvence 125kHz* [online] [cit. 2021-04-10]. Dostupné z: https://www.hadex.cz/m492a-ctecka-kodu-rfid-frekvence-125khz/.